(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 November 2014 (20.11.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/186058 A1

(51) International Patent Classification:
G06F 19/00 (2011.01) *G06F 9/44* (2006.01)

(21) International Application Number:
PCT/US2014/031921

(22) International Filing Date:
26 March 2014 (26.03.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/824,544 17 May 2013 (17.05.2013) US
14/044,417 2 October 2013 (02.10.2013) US

(71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, Redwood Shores, California 94065 (US).

(72) Inventors: **LEIGH, John**; 720 W. Capistrano Way, San Mateo, California 94402 (US). **ALLAN, David**; 1112 Blackfield Way, Mountain View, California 94040 (US). **LAU, Kwok-hung (Thomas)**; 1070 Mercedes Avenue, #25, Los Altos, California 94022 (US).

(74) Agents: **PARMENTER, Sean** et al.; Kilpatrick Townsend & Stockton LLP, Eighth Floor, Two Embarcadero Center, San Francisco, California 94111 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) Title: USE OF PROJECTOR AND SELECTOR COMPONENT TYPES FOR ETL MAP DESIGN

(57) Abstract: A data integration system is disclosed that incorporates one or more techniques for simplifying the design and maintenance of a mapping. As components are added or removed to an existing design, the data integration system removes the need to specify all input and output attributes. In one aspect, components types are implemented that allow assignment expressions to reference all or part of upstream components. Therefore, attributes of certain types of components can be propagated to downstream components or otherwise inherited from upstream components with minimal effort on the part of a map designer. During code generation the attributes required to be projected by any component can be derived based on the needs of the downstream components.

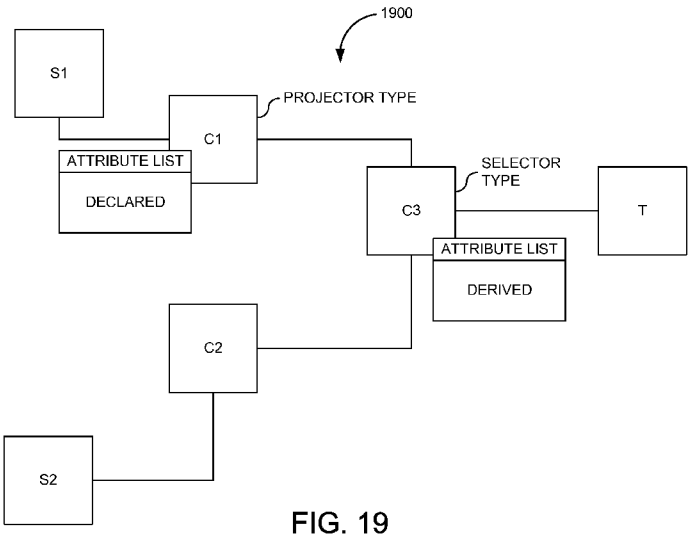

FIG. 19

WO 2014/186058 A1

USE OF PROJECTOR AND SELECTOR COMPONENT TYPES FOR ETL MAP DESIGN

BACKGROUND OF THE INVENTION

[0001] In today's increasingly fast-paced business environment, organizations need to use more specialized software applications. Additionally, organizations need to ensure the coexistence of these applications on heterogeneous hardware platforms and systems and guarantee the ability to share data between applications and systems.

[0002] Data integration is a resource-intensive procedure that requires a proprietary server having specially designed software that specifically configures it to perform data migration from one system to another. It is in these areas that typically results in poor, or inefficient performance.

[0003] Accordingly, what is desired is to solve problems relating to developing data integration scenarios, some of which may be discussed herein. Additionally, what is desired is to reduce drawbacks relating to developing data integration scenarios, some of which may be discussed herein.

BRIEF SUMMARY OF THE INVENTION

[0004] The following portion of this disclosure presents a simplified summary of one or more innovations, embodiments, and/or examples found within this disclosure for at least the purpose of providing a basic understanding of the subject matter. This summary does not attempt to provide an extensive overview of any particular embodiment or example. Additionally, this summary is not intended to identify key/critical elements of an embodiment or example or to delineate the scope of the subject matter of this disclosure. Accordingly, one purpose of this summary may be to present some innovations, embodiments, and/or examples found within this disclosure in a simplified form as a prelude to a more detailed description presented later.

[0005] In various embodiments, a data integration system enables users to create a logical design which is platform and technology independent. The user can create a logical design

that defines, at a high level, how a user wants data to flow between sources and targets. The tool can analyze the logical design, in view of the user's infrastructure, and create a physical design. The logical design can include a plurality of components corresponding to each source and target in the design, as well as operations such as joins or filters, and access points. Each component when transferred to the physical design generates code to perform operations on the data. Depending on the underlying technology (e.g., SQL Server, Oracle, Hadoop, etc.) and the language used (SQL, pig, etc.) the code generated by each component may be different.

[0006] In one aspect, a user of data integration system is not required to specify all data attributes at each component in the logical design, from start to end. The data integration system provides a plurality of component types, such as projector and selector types, that avoid the need to fully declare the information that flows through the logical design. The data integration system is able to decide what attributes are needed at operations represented by predetermined component types. This simplifies both the design and maintenance. In various aspects, data transformation and migration is provided that leverages existing RDBMS resources and capabilities to avoid the need for a separate proprietary ETL server to achieve improved performance.

[0007] In one embodiment, a method for facilitating generation of a data mapping includes receiving information specifying one or more components of a logical design, wherein at least one of the one or more components is of a first type. A set of data attributes visible to downstream components in the logical design is determined of the at least one of the one or more components that is of the first type based on upstream components in the logical design. Information indicative is then generated of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type. In one aspect, determining the set of data attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type may include deriving one or more attributes visible from an upstream component and exposing the one or more attributes to a downstream component. In another aspect, receiving the information specifying the one or more components of the logical design may include receiving information indicative of an operation that changes shape of the information flowing through the logical design.

[0008] In some embodiments, receiving the information specifying the one or more components of the logical design may include receiving information indicative of an operation that controls the flow of information flowing through the logical design but does not change shape of the information flowing through the logical design. In further embodiments, receiving the information specifying the one or more components of the logical design may include receiving information indicative of a source component having one or more attributes of data stored in a source datastore. Receiving the information specifying the one or more components of the logical design may include receiving information indicative of a target component having one or more attributes of data to be stored in a target datastore.

[0009] Generating the information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type may include exporting a list of attributes to a downstream component. In one embodiment, a change may be received in the logical design through the introduction or removal of a component or an attribute into the logical design. A determination of whether the change in the logical design affects the at least one of the one or more components that is of the first type. Based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, an updated set of data attributes visible to downstream components may be determined.

[0010] In further embodiments, a change is received in the logical design through the introduction of a component or an attribute into the logical design. A determination may be made whether the change in the logical design affects the at least one of the one or more components that is of the first type. Based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, the set of data attributes visible to downstream components may be preserved. In another aspect, based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, the set of data attributes visible to downstream components may be automatically renamed.

[0011] In one embodiment, a non-transitory computer-readable medium stores computer-executable code for facilitating generation of a data mapping. The non-transitory computer-readable medium may include code for receiving information specifying one or more components of a logical design, wherein at least one of the one or more components is of a

first type, code for determining a set of data attributes visible to downstream components in the logical design of the at least one of the one or more components that is of the first type based on upstream components in the logical design, and code for generating information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type.

[0012] In another embodiment, a system for facilitating generation of a data mapping may include a processor and a memory in communication with the processor and configured to store a set of instructions which when executed by the processor configure the processor to receive information specifying one or more components of a logical design, wherein at least one of the one or more components is of a first type, determine a set of data attributes visible to downstream components in the logical design of the at least one of the one or more components that is of the first type based on upstream components in the logical design, and generate information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type.

[0013] In one embodiment, a system for facilitating generation of a data mapping includes a receiving unit configured to receive information specifying one or more components of a logical design, wherein at least one of the one or more components is of a first type; a determining unit configured to determine a set of data attributes visible to downstream components in the logical design of the at least one of the one or more components that is of the first type based on upstream components in the logical design; and a generating unit configured to generate information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type. Determining the set of data attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type may include deriving one or more attributes visible from an upstream component and exposing the one or more attributes to a downstream component.

[0014] The information specifying the one or more components of the logical design may include information indicative of an operation that changes shape of the information flowing through the logical design. The information specifying the one or more components of the logical design may include information indicative of an operation that controls the flow of information flowing through the logical design but does not change shape of the information flowing through the logical design. The information specifying the one or more components

of the logical design may include information indicative of a source component having one or more attributes of data stored in a source datastore. The information specifying the one or more components of the logical design may include information indicative of a target component having one or more attributes of data to be stored in a target datastore. Generating the information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type may include exporting a list of attributes to a downstream component.

[0015] In one embodiment, the receiving unit is further configured to receive a change in the logical design through the introduction or removal of a component or an attribute into the logical design; the determining unit is further configured to determine whether the change in the logical design affects the at least one of the one or more components that is of the first type; and the determining unit is further configured to, based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, determine an updated set of data attributes visible to downstream components.

[0016] In one aspect, the system may further include a preserving unit, wherein the receiving unit is further configured to receive a change in the logical design through the introduction of a component or an attribute into the logical design; the determining unit is further configured to determine whether the change in the logical design affects the at least one of the one or more components that is of the first type; and the preserving unit is configured to, based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, preserve the set of data attributes visible to downstream components.

In another aspect, the system may further include a renaming unit, wherein the receiving unit is further configured to receive a change in the logical design renaming a component or an attribute; the determining unit is further configured to determine whether the change in the logical design affects the at least one of the one or more components that is of the first type; and the renaming unit is configured to, based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, rename the set of data attributes visible to downstream components.

[0017] A further understanding of the nature of and equivalents to the subject matter of this disclosure (as well as any inherent or express advantages and improvements provided) should

be realized in addition to the above section by reference to the remaining portions of this disclosure, any accompanying drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] In order to reasonably describe and illustrate those innovations, embodiments, and/or examples found within this disclosure, reference may be made to one or more accompanying drawings. The additional details or examples used to describe the one or more accompanying drawings should not be considered as limitations to the scope of any of the claimed inventions, any of the presently described embodiments and/or examples, or the presently understood best mode of any innovations presented within this disclosure.

[0019] FIG. 1 is a simplified illustration of a system that may incorporate an embodiment of the present invention.

[0020] FIG. 2 is a block diagram of a data integration system according to an embodiment of the present invention.

[0021] FIG. 3 is a simplified block diagram of a hardware/software stack that may be used to implement a data integration system according to an embodiment of the present invention.

[0022] FIG. 4 is a block diagram of an environment having various heterogeneous data sources for which data integration scenarios may be created in various embodiments of the present invention.

[0023] FIGS. 5A and 5B depict simplified data flows in conventional data integration processing that may be performed by the data integration system.

[0024] FIGS. 6A and 6B depict simplified data flows in next generation data integration processing that may be performed by the data integration system, in accordance with an embodiment of the present invention.

[0025] FIG. 7 is a simplified block diagram of interactions between an ODI Studio and a repository of the data integration system in one embodiment according to the present invention.

[0026] FIG. 8 depicts a flowchart of a method for creating a data integration scenario in accordance with an embodiment of the present invention.

[0027] FIG. 9 is a screenshot of a user interface for creating data integration scenarios in accordance with an embodiment of the present invention.

[0028] FIG. 10 depicts a flowchart of a method for creating a mapping in accordance with an embodiment of the present invention.

[0029] FIG. 11 is a screenshot of a user interface for providing mapping information in data integration scenarios in accordance with an embodiment of the present invention.

[0030] FIG. 12 is a screenshot of a user interface for providing flow information in data integration scenarios in accordance with an embodiment of the present invention.

[0031] FIG. 13 depicts a flowchart of a method for creating a package in accordance with an embodiment of the present invention.

[0032] FIG. 14 is a screenshot of a user interface for providing package sequence information in a data integration scenario in accordance with an embodiment of the present invention.

[0033] FIG. 15 depicts a flowchart of a method for deploying a data integration scenario in accordance with an embodiment of the present invention.

[0034] FIGS. 16A and 16B are simplified block diagrams of mappings in one embodiment according to the present invention.

[0035] FIG. 17 depicts a flowchart of a method for deriving component attributes based on component type in accordance with an embodiment of the present invention.

[0036] FIG. 18 is a simplified block diagram of a mapping of FIGS. 16A and 16B with a filter component in one embodiment according to the present invention.

[0037] FIG. 19 is a simplified block diagram of a mapping in one embodiment according to the present invention.

[0038] FIG. 20 depicts a flowchart of a method for generate components in accordance with an embodiment of the present invention.

[0039] FIG. 21 is a simplified block diagram of a computer system that may be used to practice embodiments of the present invention.

[0040] FIG. 22 is a simplified block diagram of a system for facilitating generation of a data mapping according to an embodiment.

DETAILED DESCRIPTION OF THE INVENTION

[0041] Introduction

[0042] In various embodiments, a data integration system enables users to create a logical design which is platform and technology independent. The user can create a logical design that defines, at a high level, how a user wants data to flow between sources and targets. The tool can analyze the logical design, in view of the user's infrastructure, and create a physical design. The logical design can include a plurality of components corresponding to each source and target in the design, as well as operations such as joins or filters, and access points. Each component when transferred to the physical design generates code to perform operations on the data. Depending on the underlying technology (e.g., SQL Server, Oracle, Hadoop, etc.) and the language used (SQL, pig, etc.) the code generated by each component may be different.

[0043] In one aspect, a user of data integration system is not required to specify all data attributes at each component in the logical design, from start to end. The data integration system provides a plurality of component types, such as projector and selector types, that avoid the need to fully declare the information that flows through the logical design. The data integration system is able to decide what attributes are needed at operations represented by predetermined component types. This simplifies both the design and maintenance.

[0044] FIG. 1 is a simplified illustration of system 100 that may incorporate an embodiment or be incorporated into an embodiment of any of the innovations, embodiments, and/or examples found within this disclosure. FIG. 1 is merely illustrative of an embodiment incorporating the present invention and does not limit the scope of the invention as recited in the claims. One of ordinary skill in the art would recognize other variations, modifications, and alternatives.

[0045] In one embodiment, system 100 includes one or more user computers 110 (e.g., computers 110A, 110B, and 110C). User computers 110 can be general purpose personal computers (including, merely by way of example, personal computers and/or laptop computers running any appropriate flavor of Microsoft Corp.'s WindowsTM and/or Apple Corp.'s MacintoshTM operating systems) and/or workstation computers running any of a variety of commercially-available UNIXTM or UNIX-like operating systems. These user computers 110 can also have any of a variety of applications, including one or more

applications configured to perform methods of the invention, as well as one or more office applications, database client and/or server applications, and web browser applications.

[0046] Alternatively, user computers 110 can be any other electronic device, such as a thin-client computer, Internet-enabled mobile telephone, and/or personal digital assistant, capable of communicating via a network (e.g., communications network 120 described below) and/or displaying and navigating web pages or other types of electronic documents. Although the exemplary system 100 is shown with three user computers, any number of user computers or devices can be supported.

[0047] Certain embodiments of the invention operate in a networked environment, which can include communications network 120. Communications network 120 can be any type of network familiar to those skilled in the art that can support data communications using any of a variety of commercially-available protocols, including without limitation TCP/IP, SNA, IPX, AppleTalk, and the like. Merely by way of example, communications network 120 can be a local area network ("LAN"), including without limitation an Ethernet network, a Token-Ring network and/or the like; a wide-area network; a virtual network, including without limitation a virtual private network ("VPN"); the Internet; an intranet; an extranet; a public switched telephone network ("PSTN"); an infra-red network; a wireless network, including without limitation a network operating under any of the IEEE 802.11 suite of protocols, the BluetoothTM protocol known in the art, and/or any other wireless protocol; and/or any combination of these and/or other networks.

[0048] Embodiments of the invention can include one or more server computers 130 (e.g., computers 130A and 130B). Each of server computers 130 may be configured with an operating system including without limitation any of those discussed above, as well as any commercially-available server operating systems. Each of server computers 130 may also be running one or more applications, which can be configured to provide services to one or more clients (e.g., user computers 110) and/or other servers (e.g., server computers 130).

[0049] Merely by way of example, one of server computers 130 may be a web server, which can be used, merely by way of example, to process requests for web pages or other electronic documents from user computers 110. The web server can also run a variety of server applications, including HTTP servers, FTP servers, CGI servers, database servers, Java servers, and the like. In some embodiments of the invention, the web server may be

configured to serve web pages that can be operated within a web browser on one or more of the user computers 110 to perform methods of the invention.

[0050] Server computers 130, in some embodiments, might include one or more file and or/application servers, which can include one or more applications accessible by a client running on one or more of user computers 110 and/or other server computers 130. Merely by way of example, one or more of server computers 130 can be one or more general purpose computers capable of executing programs or scripts in response to user computers 110 and/or other server computers 130, including without limitation web applications (which might, in some cases, be configured to perform methods of the invention).

[0051] Merely by way of example, a web application can be implemented as one or more scripts or programs written in any programming language, such as Java, C, or C++, and/or any scripting language, such as Perl, Python, or TCL, as well as combinations of any programming/scripting languages. The application server(s) can also include database servers, including without limitation those commercially available from Oracle, Microsoft, IBM and the like, which can process requests from database clients running on one of user computers 110 and/or another of server computers 130.

[0052] In some embodiments, an application server can create web pages dynamically for displaying the information in accordance with embodiments of the invention. Data provided by an application server may be formatted as web pages (comprising HTML, XML, Javascript, AJAX, etc., for example) and/or may be forwarded to one of user computers 110 via a web server (as described above, for example). Similarly, a web server might receive web page requests and/or input data from one of user computers 110 and/or forward the web page requests and/or input data to an application server.

[0053] In accordance with further embodiments, one or more of server computers 130 can function as a file server and/or can include one or more of the files necessary to implement methods of the invention incorporated by an application running on one of user computers 110 and/or another of server computers 130. Alternatively, as those skilled in the art will appreciate, a file server can include all necessary files, allowing such an application to be invoked remotely by one or more of user computers 110 and/or server computers 130. It should be noted that the functions described with respect to various servers herein (e.g., application server, database server, web server, file server, etc.) can be performed by a single

server and/or a plurality of specialized servers, depending on implementation-specific needs and parameters.

[0054] In certain embodiments, system 100 can include one or more databases 140 (e.g., databases 140A and 140B). The location of the database(s) 140 is discretionary: merely by way of example, database 140A might reside on a storage medium local to (and/or resident in) server computer 130A (and/or one or more of user computers 110). Alternatively, database 140B can be remote from any or all of user computers 110 and server computers 130, so long as it can be in communication (e.g., via communications network 120) with one or more of these. In a particular set of embodiments, databases 140 can reside in a storage-area network ("SAN") familiar to those skilled in the art. (Likewise, any necessary files for performing the functions attributed to user computers 110 and server computers 130 can be stored locally on the respective computer and/or remotely, as appropriate). In one set of embodiments, one or more of databases 140 can be a relational database that is adapted to store, update, and retrieve data in response to SQL-formatted commands. Databases 140 might be controlled and/or maintained by a database server, as described above, for example.

[0055] Data Integration Overview

[0056] FIG. 2 is a simplified block diagram of data integration system 200 according to an embodiment of the present invention. FIG. 2 is a simplified illustration of data integration system 200 that may incorporate various embodiments or implementations of the one or more inventions presented within this disclosure. FIG. 2 is merely illustrative of an embodiment or implementation of an invention disclosed herein should not limit the scope of any invention as recited in the claims. One of ordinary skill in the art may recognize through this disclosure and the teachings presented herein other variations, modifications, and/or alternatives to those embodiments or implementations illustrated in the figures.

[0057] In this embodiment, data integration system 200 includes information sources 202, information integration 204, and information destinations 206. In general, information flows from information sources 202 to information integration 204 whereby the information may be consumed, made available, or otherwise used by information destinations 206. Data flows may be unidirectional or bidirectional. In some embodiments, one or more data flows may be present in data integration system 200.

[0058] Information sources 202 are representative of one or more hardware and/or software elements configured to source data. Information sources 202 may provide direct or indirect

access to the data. In this embodiment, information sources 202 include one or more applications 208 and one or more repositories 210.

[0059] Applications 208 are representative of traditional applications, such as desktop, hosted, web-based, or cloud-based applications. Applications 208 may be configured to receive, process, and maintain data for one or more predetermined purposes. Some examples of applications 208 include customer relationship management (CRM) applications, financial services applications, government and risk compliance applications, human capital management (HCM), procurement applications, supply chain management applications, project or portfolio management applications, or the like. Applications 208 may include functionality configured for manipulating and exporting application data in a variety of human-readable and machine-readable formats, as is known in the art. Applications 208 may further access and store data in repositories 210.

[0060] Repositories 210 are representative of hardware and/or software elements configured to provide access to data. Repositories 210 may provide logical and/or physical partitioning of data. Repositories 210 may further provide for reporting and data analysis. Some examples of repositories 210 include databases, data warehouses, cloud storage, or the like. A repository may include a central repository created by integrating data from one or more applications 208. Data stored in repositories 210 may be uploaded from an operational system. The data may pass through additional operations before being made available in a source.

[0061] Information integration 204 is representative of one or more hardware and/or software elements configured to provide data integration services. Direct or indirect data integration services can be provided in information integration 204. In this embodiment, information integration 204 includes data migration 212, data warehousing 214, master data management 216, data synchronization 218, federation 220, and real-time messaging 222. It will be understood that information integration 204 can include one or more modules, services, or other additional elements than those shown in here that provide data integration functionality.

[0062] Data migration 212 is representative of one or more hardware and/or software elements configured to provide data migration. In general, data migration 212 provides one or more processes for transferring data between storage types, formats, or systems. Data migration 212 usually provides for manual or programmatic options to achieve a migration.

In a data migration procedure, data on or provided by one system is mapped to another system providing a design for data extraction and data loading. A data migration may involve one or more phases, such a design phase where one or more designs are created that relate data formats of a first system to formats and requirements of a second system, a data extraction phase where data is read from the first system, a data cleansing phase, and a data loading phase where data is written to the second system. In some embodiments, a data migration may include a data verification phases to determine whether data is accurately processed in any of the above phases.

[0063] Data warehousing 214 is representative of one or more hardware and/or software elements configured to provide databases used for reporting and data analysis. A data warehouse is typically viewed as a central repository of data which is created by integrating data from one or more disparate sources. Data warehousing 214 may include the current storage of data as well as storage of historical data. Data warehousing 214 may include typical extract, transform, load (ETL)-based data warehouse whereby staging, data integration, and access layers house key functions. In one example, a staging layer or staging database stores raw data extracted from each of one or more disparate source data systems. An integration layer integrates disparate data sets by transforming the data from the staging layer often storing this transformed data in an operational data store (ODS) database. The integrated data is then moved to yet another database, often called the data warehouse database. The data can be arranged into hierarchical groups (often called dimensions) and into facts and aggregate facts. An access layer may be provided to help users or other systems retrieve data. Data warehouses can be subdivided into data marts whereby each data mart stores subsets of data from a warehouse. In some embodiments, data warehousing 214 may include business intelligence tools, tools to extract, transform and load data into the repository, and tools to manage and retrieve metadata.

[0064] Master data management 216 is representative of one or more hardware and/or software elements configured to manage a master copy of data. Master data management 216 may include a set of processes, governance, policies, standards and tools that consistently define and manage master data. Master data management 216 may include functionality for removing duplicates, standardizing data, and incorporating rules to eliminate incorrect data from entering a system in order to create an authoritative source of master data. Master data management 216 may provide processes for collecting, aggregating, matching, consolidating,

quality-assuring, persisting and distributing data throughout an organization to ensure consistency and control in the ongoing maintenance and application use of information.

[0065] Data synchronization 218 is representative of one or more hardware and/or software elements configured to synchronize data. Data synchronization 218 may provide for establishing consistency among data from a source to a target and vice versa. Data synchronization 218 may further provide for the continuous harmonization of the data over time.

[0066] Federation 220 is representative of one or more hardware and/or software elements configured to consolidate a view of data from constituent sources. Federation 220 may transparently map multiple autonomous database systems into a single federated database. The constituent databases maybe interconnected via a computer network and may be geographically decentralized. Federation 220 provides an alternative to merging several disparate databases. A federated database, or virtual database, for example, may provide a composite of all constituent databases. Federation 220 may not provide actual data integration in the constituent disparate databases but only in the view.

[0067] Federation 220 may include functionality that provides a uniform user interface, enabling users and clients to store and retrieve data in multiple noncontiguous databases with a single query -- even if the constituent databases are heterogeneous. Federation 220 may include functionality to decompose a query into subqueries for submission to relevant constituent data sources and composite the result sets of the subqueries. Federation 220 can include one or more wrappers to the subqueries to translate them into appropriate query languages. In some embodiments, federation 220 is a collection of autonomous components that make their data available to other members of the federation through the publication of an export schema and access operations.

[0068] Real-time messaging 222 is representative of one or more hardware and/or software elements configured to provide messaging services subject to a real-time constraint (e.g., operational deadlines from event to system response). Real-time messaging 222 may include functionality that guarantees an action or response within strict time constraints. In one example, real-time messaging 222 may be tasked with taking some orders and customer data from one database, combining it with some employee data held in a file, and then loading the integrated data into a Microsoft SQL Server 2000 database. Because orders need to be analyzed as they arrive, real-time messaging 222 may pass the orders through to a target

database in as close to real time as possible and extract only the new and changed data to keep the workload as small as possible.

[0069] Information destinations 206 are representative of one or more hardware and/or software elements configured to store or consume data. In this embodiment, information destinations 206 may provide direct or indirect access to the data. In this embodiment, information destinations 206 include one or more applications 224 and one or more repositories 226.

[0070] Applications 224 are representative of traditional applications, such as desktop, hosted, web-based, or cloud-based applications. Applications 224 may be configured to receive, process, and maintain data for one or more predetermined purposes. Some examples of applications 224 include customer relationship management (CRM) applications, financial services applications, government and risk compliance applications, human capital management (HCM), procurement applications, supply chain management applications, project or portfolio management applications, or the like. Applications 224 may include functionality configured for manipulating and importing application data in a variety of human-readable and machine-readable formats, as is known in the art. Applications 224 may further access and store data in repositories 226.

[0071] Repositories 226 are representative of hardware and/or software elements configured to provide access to data. Repositories 226 may provide logical and/or physical partitioning of data. Repositories 226 may further provide for reporting and data analysis. Some examples of repositories 226 include databases, data warehouses, cloud storage, or the like. A repository may include a central repository created by integrating data from one or more applications 226. Data stored in repositories 226 may be uploaded or imported through information integration 204. The data may pass through additional operations before being made available at a destination.

[0072] Data Integration System

[0073] FIG. 3 is a simplified block diagram of a hardware/software stack that may be used to implement data integration system 200 according to an embodiment of the present invention. FIG. 3 is merely illustrative of an embodiment or implementation of an invention disclosed herein should not limit the scope of any invention as recited in the claims. One of ordinary skill in the art may recognize through this disclosure and the teachings presented herein other variations, modifications, and/or alternatives to those embodiments or

implementations illustrated in the figures. One example of components found within data integration system 200 according to this embodiment may include ORACLE DATA INTEGRATOR, a member of the ORACLE FUSION Middleware family of products provided by Oracle of Redwood Shores, California. ORACLE DATA INTEGRATOR is a Java-based application that uses one or more databases to perform set-based data integration tasks. In addition, ORACLE DATA INTEGRATOR can extract data, provide transformed data through Web services and messages, and create integration processes that respond to and create events in service-oriented architectures. ORACLE DATA INTEGRATOR is based on an ELT [extract-Load and Transform] architecture rather than conventional ETL [extract-transform-load] architectures. A copy of a user manual for ORACLE DATA INTEGRATOR is attached to this disclosure and incorporated herein by reference for all purposes.

[0074] In various embodiments, data integration system 200 provides a new declarative design approach to defining data transformation and integration processes, resulting in faster and simpler development and maintenance. Data integration system 200 thus separates declarative rules from the implementation details. Data integration system 200 further provides a unique E-LT architecture (Extract - Load Transform) for the execution of data transformation and validation processes. This architecture in embodiments eliminates the need for a standalone ETL server and proprietary engine. In some embodiments, data integration system 200 instead leverages the inherent power of RDBMS engines.

[0075] In some embodiments, data integration system 200 integrates in one or more middleware software packages, such as the ORACLE FUSION MIDDLEWARE platform and becomes a component of the middleware stack. As depicted in FIG. 3 data integration system 200 may provide run-time components as Java EE applications.

[0076] In this example, one component of data integration system 200 is repositories 302. Repositories 302 are representative of hardware and/or software elements configured to store configuration information about an IT infrastructure, metadata of all applications, projects, scenarios, and execution logs. In some aspects, multiple instances of repositories 302 can coexist in an IT infrastructure, for example Development, QA, User, Acceptance, and Production. Repositories 302 are configured to allow several separated environments that exchange metadata and scenarios (for example: Development, Test, Maintenance and Production environments). Repositories 302 further are configured to act as a version control system where objects are archived and assigned a version number.

[0077] In this example, repositories 302 are composed of at least one master repository 304 and one or more work repositories 306. Objects developed or configured for use within data integration system 200 may be stored in one of these repository types. In general, master repository 304 stores the following information: security information including users, profiles and rights, topology information including technologies, server definitions, schemas, contexts, languages and so forth, and versioned and archived objects. The one or more work repositories 306 may contain actual developed objects.

[0078] Several work repositories may coexist in data integration system 200 (for example, to have separate environments or to match a particular versioning life cycle). The one or more work repositories 306 store information for models, including schema definition, data stores structures and metadata, fields and columns definitions, data quality constraints, cross references, data lineage, and so forth. The one or more work repositories 306 may further store projects, including business rules, packages, procedures, folders, knowledge modules, variables and so forth, and scenario execution, including scenarios, scheduling information and logs. In some aspects, the one or more work repositories 306 may contain only execution information (typically for production purposes), and be designated as an execution repository.

[0079] In various embodiments, repositories 302 store one or more ETL projects. An ETL project defines or otherwise specifies one or more data models that model data attributes of data in a source or target. An ETL project further provides for data quality control as well as defining mappings to move and transform data. Data integrity control ensures the overall consistency of the data. Application data is not always valid for the constraints and declarative rules imposed by a particular source or target. For example, orders may be found with no customer, or order lines with no product, and so forth. Data integration system 200 provides a working environment to detect these constraint violations and to store them for recycling or reporting purposes.

[0080] In some embodiments of data integration system 200, there are two different types of controls: Static Control and Flow Control. Static Control implies the existence of rules that are used to verify the integrity of application data. Some of these rules (referred to as constraints) may already be implemented in data servers (using primary keys, reference constraints, etc.). Data integration system 200 allows for the definition and checking of additional constraints, without declaring them directly in a source. Flow Control relates to targets of transformation and integration processes that implement their own declarative

rules. Flow Control verifies an application's incoming data according to these constraints before loading the data into a target. Flow control procedures are generally referred to as mappings.

[0081] An ETL project can be automated into a package that can be deployed for execution in a runtime environment. Accordingly, the automation of data integration flows is achieved by sequencing the execution of the different steps (mappings, procedures, and so forth) in a package and by producing a production scenario containing ready-to-use code for each of these steps. A package is typically made up of a sequence of steps organized into an execution diagram. Packages are the main objects used to generate scenarios for production. They represent the data integration workflow and can perform jobs, such as for example: start a reverse-engineering process on a datastore or a model, send an email to an administrator, download a file and unzip it, define the order in which mappings must be executed, and define loops to iterate over execution commands with changing parameters.

[0082] A scenario is designed to put a source component (mapping, package, procedure, variable) into production. A scenario results from the generation of code (SQL, shell, and so forth) for this component. Once generated, the code of the source component is frozen and the scenario is stored inside repositories 302, such as one or more of work repositories 306. A scenario can be exported and then imported into different production environments.

[0083] In various embodiments, data integration system 200 is organized around repositories 302 in a modular fashion accessed by Java graphical modules and scheduling agents. Graphical modules can be used to design and build one or more integration processes stored in repositories 302. Administrators, Developers and Operators may use a development studio to access repositories 302. Agents can be used to schedule and coordinate a set of integration tasks associated with an integration process stored in repositories 302. For example, at runtime, an agent deployed on a desktop, web services, or otherwise in communication with a source coordinates the execution of one or more integration processes. The agent may retrieve code stored in master repository 304, connect to various source and target systems, and orchestrate an overall data integration process or scenario.

[0084] In this embodiment, data integration system 200 includes desktop 308 that may include one or more of the above discussed graphical modules and/or agents. Desktop 308 is representative of one or more desktop or workstation computing devices, such as personal computers, laptops, netbooks, tablets, and the like. Desktop 308 includes a Java virtual

machine (JVM) 310 and Oracle Data Integrator (ODI) Studio 312. Java virtual machine (JVM) 310 is a virtual machine that can execute Java bytecode. JVM 310 is most often implemented to run on an existing operating system, but can also be implemented to run directly on hardware. JVM 310 provides a run-time environment in which Java bytecode can be executed, enabling features such as runtime web service (WS) 314 and agent 316. JVM 310 may include a Java Class Library, a set of standard class libraries (in Java bytecode) that implement the Java application programming interface (API), and other elements that form a Java Runtime Environment (JRE).

[0085] Agent 316 is configured to schedule and coordinate a set of integration tasks associated with one or more integration processes stored in repositories 302. For example, at runtime, an agent coordinates the execution of integration processes. The agent may retrieve code stored in master repository 304, connect to various source and target systems, and orchestrate an overall data integration process or scenario.

[0086] Referring again to FIG. 3, ODI Studio 312 includes hardware and/or software elements configured to design data integration projects. In this example, ODI Studio 312 includes four graphical modules or navigators that are used to create and manage data integration projects, namely, designer module 318, operator module 320, topology module 322, and security module 324. Designer module 318 is a module configured to define data stores (tables, files, Web services, and so on), data mappings, and packages (sets of integration steps, including mappings). In various embodiments, designer module 318 defines declarative rules for data transformation and data integrity. Accordingly, project development takes place in designer module 318. Additionally, in designer module 318, is where database and application metadata are imported and defined. Designer module 318, in one embodiment, uses metadata and rules to generate data integration scenarios or load plans for production. In general, designer module 318 is used to design data integrity checks and to build transformations such as for example: automatic reverse-engineering of existing applications or databases, graphical development and maintenance of transformation and integration mappings, visualization of data flows in the mappings, automatic documentation generation, and customization of generated code.

[0087] Operator module 320 is a module configured to view and manage production integration jobs. Operator module 320, thus, manages and monitors data integration processes in production and may show execution logs with error counts, the number of rows

processed, execution statistics, the actual code that is executed, and so on. At design time, developers can also use operator module 320 for debugging purposes in connection with designer module 318.

[0088] Topology module 322 is a module configured to create and manage connections to datasources and agents. Topology module 322 defines the physical and logical architecture of the infrastructure. Infrastructure or projects administrators may register servers, database schemas and catalogs, and agents in a master repository through topology module 322. Security module 324 is a module configured to manage users and their repository privileges.

[0089] In general, a user or process interacts with designer module 318 to create a data integration project having one or more data integration processes for sources and targets 326. Each data integration process includes at least one data integration task. In some embodiments, a data integration task is defined by a set of business rules indicative of what bit of data is to be transformed and combined with other bits as well as technical specifics of how the data is actually extracted, loaded, and so on. In preferred embodiments, a data integration task is specified using a declarative approach to build data mappings. A mapping is an object that populates one datastore, called the target, which data coming from one or more other datastores, known as sources. In general, columns in the source datastore are linked to the columns in the target datastore through mapping. A mapping can be added into a package as a package step. As discussed above, a package defines a data integration job. A package is created under a project and is made up of an organized sequence of steps, each of which can be a mapping or a procedure. A package can have one entry point and multiple exit points.

[0090] In some embodiments, when creating a new mapping, a developer or technical business user interacts with designer 318 to first define which data is integrated and which business rules should be used. For example, the developer may specify what tables are to be joined, filters to be applied, and SQL expressions to be used to transform data. The particular dialect of SQL that is used is determined by the database platform on which the code is to be executed. Then, in a separate step, technical staff can interact with designer 318 to choose the most efficient way to extract, combine, and then integrate this data. For example, the technical staff may use database-specific tools and design techniques such as incremental loads, bulk-loading utilities, slowly changing dimensions, and changed-data capture.

[0091] In this embodiment, mappings can be created for sources and targets 326. Sources and targets 326 may include one or more legacy applications 328, one or more files/XML documents 330, one or more applications 332, one or more data warehouses (DW), business intelligence (BI) tools and applications, and enterprise process management (EPM) tools and applications 334, and one or more JVMs 336 (including runtime web service 340 and agent 342).

[0092] FIG. 4 is a block diagram of environment 400 having various heterogeneous data sources for which data integration scenarios may be created in various embodiments of the present invention. In this example, environment 400 includes ODI Studio 312 and repositories 302. Repositories 302 contain all of the metadata required to generate integration scenarios 400. A user or process interacts with ODI Studio 312 to create integration scenarios 400 using data integrity controls 402 and declarative rules 404.

[0093] Orders application 406 is representative of an application for tracking customer orders. An "Orders Application" data model is created to represent data stored in Orders application 406 as well as any data integrity controls or conditions. For example, the "Orders Application" data model may be based on a Hyper Structured Query Language Database (HSQLDB) mapping and include five datastores, SRC_CITY, SRC_CUSTOMER, SRC_ORDERS, SRC_ORDER_LINES, SRC_PRODUCT, and SRC_REGION.

[0094] Parameter file 408 is representative of a flat file (e.g., ASCII) issued from a production system containing a list of sales representatives and the segmentation of ages into age ranges. In this example, a "Parameter" data model is created to represent the data in the flat file. For example, the "Parameter" data model may be based on a file interface and include two datastores, SRC_SALES _PERSON and SRC_AGE_GROUP.

[0095] Sales administration application 410 is representative of an application for tracking sales. The sales administration application 410 may be a data warehouse populated with transformations of data from orders application 406 and parameter file 408. A "Sales Administration" data model is created to represent data stored in sales administration application 410 as well as any data integrity controls or conditions or transformations. For example, the "Sales Administration" data model may be based on a Hyper Structured Query Language Database (HSQLDB) mapping and include six datastores, TRG_CITY, TRG_COUNTRY, TRG_CUSTOMER, TRG_PRODUCT, TRG_PROD_FAMILY, TRG_REGION, and TRG_SALE.

[0096] FIGS. 5A and 5B depict simplified data flows in conventional data integration processing that may be performed by data integration system 200. In this example, data from orders application 406, parameter file 408, and one or more other optional or additional sources flow through a traditional ETL process targeted to sales administration application 410. Data transforms occur in a separate ETL server 500. The scenario requires dedicated or proprietary resources, results in poorer performance, and incurs high costs.

[0097] FIGS. 6A and 6B depict simplified data flows in next generation data integration processing that may be performed by data integration system 200, in accordance with an embodiment of the present invention. In this example, data from orders application 406, parameter file 408, and one or more other optional or additional sources flow through E-LT process targeted to sales administration application 410. Data transforms leverage existing resources resulting in higher performance and efficiency. As described above, prior ETL systems required dedicated and/or proprietary infrastructure to perform data transforms. This was done, in part, to accommodate unknown user infrastructures. For example, without knowing what types of databases are being used, prior ETL systems were unable to anticipate what transform operations would be available in a given system. However, this results in under-utilized resources, such as the user's existing databases and servers which are capable of executing the appropriate data transforms without any dedicated and/or proprietary infrastructure.

[0098] In accordance with an embodiment, the present invention leverages the user's existing infrastructure by enabling the user to customize a data integration process according to the user's particular needs. For example, when a data integration plan is designed, it can be divided into discrete portions which are executable by a single system, referred to as execution units. Once a data integration plan has been divided into a plurality of execution units, the user can be presented with a physical plan based on the user's infrastructure and system resources. This plan can be further customized by the user to change which user systems execute which execution units. For example, a user may be presented with a plan in which a join operation is executed on a first database, and the user may customize the plan by moving the join operation to a second database.

[0099] As shown in FIG. 6B, this results in an extract-load-transform (E-LT) architecture that does not rely on a stand-alone transform server which characterized prior ETL systems. Instead, as described above, data transforms can be performed on the user's existing

infrastructure. The E-LT architecture provides users with greater flexibility while reducing costs associated with acquiring and maintaining proprietary transform servers.

[0100] Referring again to FIG. 3, agents can be used to schedule and coordinate a set of integration tasks associated with an integration process. For example, at runtime, an agent coordinates the execution of integration processes. The agent may retrieve code stored in master repository 304, connect to the various source and target systems and orchestrates an overall data integration process or scenario. In various embodiments, there are two types of agents. In one example, a standalone agent is installed on desktop 308, such as agent 316. In another example, an application server agent can be deployed on application server 326 (such as a Java EE Agent deployed on an Oracle WebLogic Server) and can benefit from the application server layer features such as clustering for High Availability requirements. In yet another example, an agent can be deployed on sources and targets 326, such as agent 342.

[0101] In this embodiment, data integration system 200 includes application server 344 that may include one or more of the above discussed agents. Application server 344 is representative of one or more application servers, web-servers, or hosted applications. In this example, application server 344 includes FMW console 346, servlet container 348, web services container 350, and data sources connection pool 352.

[0102] FMW console 346 is representative of one or more hardware and/or software elements configured to manage aspects of application server 344, such as information related to servlet container 348, web services container 350, and data sources connection pool 334. For example, FMW console 346 may be a browser-based, graphical user interface used to manage an Oracle WebLogic Server domain. FMW console 346 may include functionality to configure, start, and stop WebLogic Server instances, configure WebLogic Server clusters, configure WebLogic Server services, such as database connectivity (JDBC) and messaging (JMS), configure security parameters, including creating and managing users, groups, and roles, configure and deploy Java EE applications, monitor server and application performance, view server and domain log files, view application deployment descriptors, and edit selected run-time application deployment descriptor elements. In some embodiments, FMW console 346 includes ODI plug-in 354 providing FMW console 346 with access to data integration processes in production and may show execution logs with error counts, the number of rows processed, execution statistics, the actual code that is executed, and so forth.

[0103] Servlet container 348 is representative of one or more hardware and/or software elements configured to extend the capabilities of application server 344. Servlets are most often used to process or store data that was submitted from an HTML form, provide dynamic content such as the results of a database query, and manage state information that does not exist in the stateless HTTP protocol, such as filling the articles into the shopping cart of the appropriate customer. A servlet is typically a Java class in Java EE that conforms to the Java Servlet API, a protocol by which a Java class may respond to requests. To deploy and run a servlet, servlet container 348 is used as a component of a web server that interacts with servlets. Accordingly, servlet container 348 may extend functionality provided by public web service 356 and data services 358 of web services container 350 as well as access to data pools provided by data sources connection pool 352. Servlet container 348 is also responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

[0104] In this example, servlet container 348 includes Java EE application 360 associated with ODI SDK 362, ODI console 364, and runtime web service 366 associated with Java EE agent 368. ODI SDK 362 provides a software development kit (SDK) for data integration and ETL design. ODI SDK 362 enables automation of work that is common and very repetitive allowing a user to script repetitive tasks.

[0105] ODI console 364 is a Java Enterprise Edition (Java EE) application that provides Web access to repositories 302. ODI console 364 is configured to allow users to browse Design-Time objects, including projects, models, and execution logs. ODI console 364 may allow users to view flow maps, trace the source of all data, and even drill down to the field level to understand the transformations used to build the data. In addition, end users can launch and monitor scenario execution through ODI console 364. In one aspect, ODI console 364 provides administrators with the ability to view and edit Topology objects such as Data Servers, Physical and Logical Schemas as well as to manage repositories 302.

[0106] Data Scenario Design and Development

[0107] As discussed above, a scenario is designed to put a source component (mapping, package, procedure, variable) into production. A scenario results from the generation of code (SQL, shell, and so forth) for this component. A scenario can be exported and then imported into different production environments.

[0108] FIG. 7 is a simplified block diagram of interactions between an ODI Studio and a repository of the data integration system in one embodiment according to the present invention. In the embodiment shown in FIG. 7, ODI Studio 312 of FIG. 3 uses metadata and rules to generate data integration scenarios 700 for production. In general, designer module 318 is used to design data integrity checks and to build transformations such as for example: automatic reverse-engineering of existing applications or databases, graphical development and maintenance of transformation and integration mappings, visualization of data flows in the mappings, automatic documentation generation, and customization of generated code.

[0109] FIG. 8 depicts a flowchart of method 800 for creating a data integration scenario in accordance with an embodiment of the present invention. Implementations of or processing in method 800 depicted in FIG. 8 may be performed by software (e.g., instructions or code modules) when executed by a central processing unit (CPU or processor) of a logic machine, such as a computer system or information processing device, by hardware components of an electronic device or application-specific integrated circuits, or by combinations of software and hardware elements. Method 800 depicted in FIG. 8 begins in step 810.

[0110] In various embodiments, a user may initiate a session with designer module 318 of ODI Studio 312 and connect to repositories 302. The user may interact with one or more user interface features to create a new data integration project or select from existing data integration projects stored in, for example, master repository 304. In general, designer module 318 is used to manage metadata, to design data integrity checks, and to build transformations. In various embodiments, the main objects handled through designer module 318 are models and projects. Data models contain all of the metadata in a data source or target (e.g., tables, columns, constraints, descriptions, cross-references, etc.). Projects contain all of the loading and transformation rules for a source or target (e.g., mappings, procedures, variables, etc.).

[0111] In step 820, one or more data models are created. In step 830, one or more projects are created. FIG. 9 is a screenshot of user interface 900 for creating a data integration scenario in accordance with an embodiment of the present invention. In this example, navigation panel 910 displays information and includes functionality for interacting with projects. Navigation panel 920 displays information and includes functionality for interacting with data models. As discussed above, the user may not only create the data model, but also develop any data integrity checks for the data in the data models. Additionally, the user may

specify mappings, procedures, variables for projects that provide data integrity and transforms for the data in a flow that loads data from a source into a target. In step 840, one or more data integration scenarios are generated. FIG. 8 ends in step 850.

[0112] FIG. 10 depicts a flowchart of method 1000 for creating a mapping in accordance with an embodiment of the present invention. Implementations of or processing in method 1000 depicted in FIG. 10 may be performed by software (e.g., instructions or code modules) when executed by a central processing unit (CPU or processor) of a logic machine, such as a computer system or information processing device, by hardware components of an electronic device or application-specific integrated circuits, or by combinations of software and hardware elements. Method 1000 depicted in FIG. 10 begins in step 1010.

[0113] In step 1020, target datastore information is received. For example, a user may interact with one or more user interface features of designer module 318 to provide target datastore information. In one embodiment, the user may drag and drop target datastore information comprising one or more data models from navigation panel 910 onto a mapping or flow panel that visually represents aspects of a selected data model and any associated transforms or data integrity checks.

[0114] In step 1030, source datastore information is received. For example, a user may interact with one or more user interface features of designer module 318 to provide source datastore information. In one embodiment, the user may drag and drop source datastore information comprising one or more data models from navigation panel 910 onto the same mapping or flow panel of the target datastore information that visually represents aspects of a selected data model and any associated transforms or data integrity checks.

[0115] In various embodiments, the source datastore information and the target data store information may be composed of one or more data models and optionally operations. Some examples of operations can include one or more data set operations (e.g., unions, joins, intersections, etc.), data transformations, data filter operations, constraints, descriptions, cross-references, integrity checks, or the like. In further embodiments, some of these operations may be preconfigured and visually represented in designer module 318. In other embodiments, custom operations may be provided allowing the user to specify logic, mappings, and the like that implement an operation.

[0116] In step 1040, mapping information is received. For example, a user may interact with one or more user interface features of designer module 318 to map the source datastore

information to the target datastore information. In one embodiment, the user may visually connect attributes of data elements in the source datastore information with attributes of data elements in the target datastore information. This may be done by matching column names of tables in the source datastore information and the target datastore information. In further embodiments, one or more automatic mapping techniques may be used to provide mapping information.

[0117] FIG. 11 is a screenshot of user interface 1100 for providing mapping information in a data integration scenario in accordance with an embodiment of the present invention. In this example, attributes of source components are mapped to attributes of target components.

[0118] Referring again to FIG. 10, in step 1050, data loading strategies are received. A data loading strategy includes information on how the actual data from the source datastore information is to be loaded during an extract phase. Data loading strategies can be defined in a flow tab of designer 318. In some embodiments, a data loading strategy can be automatically computed for a flow depending on a configuration of the mapping.

[0119] For example, one or more knowledge modules may be proposed for the flow. A knowledge module (KM) is a component that implements reusable transformation and ELT (extract, load, and transform) strategies across different technologies. In one aspect, knowledge modules (KMs) are code templates. Each KM can be dedicated to an individual task in an overall data integration process. The code in KMs appears in nearly the form that it will be executed with substitution methods enabling it to be used generically by many different integration jobs. The code that is generated and executed is derived from the declarative rules and metadata defined in the designer module 318. One example of this is extracting data through change data capture from Oracle Database 10g and loading the transformed data into a partitioned fact table in Oracle Database 11g, or creating timestamp-based extracts from a Microsoft SQL Server database and loading this data into a Teradata enterprise data warehouse.

[0120] The power of KMs lies in their reusability and flexibility—for example, a loading strategy can be developed for one fact table and then the loading strategy can be applied to all other fact tables. In one aspect, all mappings that use a given KM inherit any changes made to the KM. In some embodiments, five different types of KMs are provided, each of them covering one phase in a transformation process from source to target, such as an integration

knowledge module (IKM), a loading knowledge module (LKM), and a check knowledge module CKM.

[0121] Referring to FIG. 4, a user may define a way to retrieve the data from SRC_AGE_GROUP, SRC_SALES_PERSON files and from the SRC_CUSTOMER table in environment 400. To define a loading strategies, a user may select a source set that corresponds to the loading of the SRC_AGE_GROUP file and select a LKM File to SQL to implement the flow from a file to SQL. In one aspect, a LKM is in charge of loading source data from a remote server to a staging area.

[0122] In step 1060, data integration strategies are received. After defining the loading phase, the user defines a strategy to adopt for the integration of the loaded data into a target. To define the integration strategies, the user may select a target object and select an IKM SQL Incremental Update. An IKM is in charge of writing the final, transformed data to a target. When an IKM is started, it assumes that all loading phases for remote servers have already carried out their tasks, such as having all remote source data sets loaded by LKMs into a staging area, or the source datastores are on the same data server as the staging area.

[0123] In step 1070, data control strategies are received. In general, a CKM is in charge of checking that records of a data set are consistent with defined constraints. A CKM may be used to maintain data integrity and participates in overall data quality initiative. A CKM can be used in 2 ways. First, to check the consistency of existing data. This can be done on any datastore or within mappings. In this case, the data checked is the data currently in the datastore. In a second case, data in the target datastore is checked after it is loaded. In this case, the CKM simulates the constraints of the target datastore on the resulting flow prior to writing to the target.

[0124] FIG. 12 is a screenshot of user interface 1200 for providing flow information in a data integration scenario in accordance with an embodiment of the present invention.

[0125] In step 1080, a mapping is generated. FIG. 10 ends in step 1090.

[0126] Data Integration Scenario Packages and Deployment

[0127] As discussed above, automation of data integration flows can be achieved in data integration system 200 by sequencing the execution of the different steps (mappings, procedures, and so forth) in a package and by producing a production scenario containing the ready-to-use code for each of these steps. A package is made up of a sequence of steps

organized into an execution diagram. Packages are the main objects used to generate scenarios for production. A scenario is designed to put a source component (mapping, package, procedure, variable) into production. A scenario results from the generation of code (SQL, shell, and so forth) for this component. A scenario can be exported and then imported into different production environments.

[0128] FIG. 13 depicts a flowchart of a method for creating a package in accordance with an embodiment of the present invention. Implementations of or processing in method 1300 depicted in FIG. 13 may be performed by software (e.g., instructions or code modules) when executed by a central processing unit (CPU or processor) of a logic machine, such as a computer system or information processing device, by hardware components of an electronic device or application-specific integrated circuits, or by combinations of software and hardware elements. Method 1300 depicted in FIG. 13 begins in step 1310.

[0129] In step 1320, package step information is received. The package step information includes information identifying a step, elements, properties, components, and the like. In one example, a user may interact with one or more user interface features of designer module 318 to create, identify, or otherwise specify one or more steps for a package. In one embodiment, one or more components are selected and placed on a diagram. These components appear as steps in the package.

[0130] In step 1330, package step sequence information is received. The package step sequence information includes information identifying an ordering for a step, dependencies, and the like. Once steps are created, the steps are ordered or reordered into a data processing chain. In one example, a user may interact with one or more user interface features of designer module 318 to provide sequencing or ordering for one or more steps of a package. A data processing chain may include a unique step defined as a first step. Generally, each step has one or more termination states, such as success or failure. A step in some states, such as failure or success, can be followed by another step or by the end of the package. In one aspect, in case of some states, such as failure, sequence information may define a number of retries. In another aspect, a package may have but several possible termination steps.

[0131] FIG. 14 is a screenshot of a user interface for providing package step sequence information in a data integration scenario in accordance with an embodiment of the present invention.

[0132] In step 1340, a package is generated. FIG. 13 ends in step 1350.

[0133] As discussed above, the automation of data integration flows can be achieved by sequencing the execution of different steps (mappings, procedures, and so forth) in a package. The package can then be produced for a production scenario containing the ready-to-use code for each of the package's steps. In various embodiments, the package is deployed to run automatically in a production environment.

[0134] FIG. 15 depicts a flowchart of method 1500 for deploying a data integration scenario in accordance with an embodiment of the present invention. Implementations of or processing in method 1500 depicted in FIG. 15 may be performed by software (e.g., instructions or code modules) when executed by a central processing unit (CPU or processor) of a logic machine, such as a computer system or information processing device, by hardware components of an electronic device or application-specific integrated circuits, or by combinations of software and hardware elements. Method 1500 depicted in FIG. 15 begins in step 1510.

[0135] In step 1520, an integration scenario is retrieved. In one embodiment, a package is retrieved from repositories 302. In step 1530, the integration scenario is deployed to one or more agents. In step 1540, the integration scenario is executed by the one or more agents. In one aspect, the integration scenario can be executed in several ways, such as from ODI Studio 312, from a command line, or from a web service. Scenario execution can be viewed and monitored, for example, via operator module 320 and the like as discussed above. FIG. 15 ends in step 1550.

[0136] Use of Projector and Selector Types

[0137] In most data integration systems, a mapping requires an explicit definition of all input and output attributes that form part of a map. In typical flow based ETL tools, connectors are made at the attribute level. This results in a very concise mapping model. However, this also generates a huge number of objects and makes constructing and maintaining maps cumbersome due to the number of attribute level connectors.

[0138] In various embodiments, data integration system 200 incorporates one or more techniques for easing the design and maintenance of a mapping. Components can simply be added to an existing design without the need to specify all input and output attributes and allowing component level connectors to be rerouted. In one aspect, components are implemented that allow assignment expressions to reference all or part of upstream components. Accordingly, attributes of certain types of components can be propagated to

downstream components or otherwise inherited from upstream components with minimal effort on the part of a mapping designer.

[0139] In one aspect, data integration system 200 minimizes the need to manage attribute level connectivity. For example, data integration system 200 may classify components of a mapping (such as Joiner, Set, Table, Filter, etc.). Some examples of categories are projectors and selectors. In general, projectors are components that influence the shape of the data that flows through a map. Selectors are components that control the flow of the data, but don't fundamentally change the shape of the flow. In various embodiments, selector type components are configured to be transparent in that attributes of upstream components are visible. Projector type components are configured to be opaque that attributes of upstream components are not visible, only those made available in the shape of the data flow.

[0140] Accordingly, requiring each attribute to be connected to a downstream component's attributes in order to be able to see the data from upstream, in various embodiments, data integration system 200 enables users to directly reference any upstream attribute up to and including the closest projector's attributes. Therefore, data integration system 200 greatly eases the design and maintenance of a mapping. Data integration system 200 further makes adding in components to an existing design simple, typically just needing component level connectors to be rerouted.

[0141] FIGS. 16A and 16B are simplified block diagrams of mappings in one embodiment according to the present invention. In this example, mapping 1600 of FIG. 16A includes component 1610 representing a data source SRC_EMP, component 1620 representing data source SRC_DEPT, and component 1630 representing data target TGT_EMPDEPT. In order to update data target TGT_EMPDEPT, a join is needed for data source SRC_EMP and SRC_DEPT. Component 1640 representing a JOIN is added to mapping 1600 that connects to components 1610 and 1620 as inputs and to component 1630 as output. Component 1640 is configured to provide a join expression, such as (SRC_EMP.DEPTNO = SRC_DEPT.DEPTNO).

[0142] In traditional data integration systems, mapping 1600 requires an explicit definition of all input and output attributes that form part of component 1640 representing the JOIN. In contrast, in various embodiments, a mapping developer can define how columns of data target TGT_EMPDEPT are populated directly from attributes of data source SRC_EMP represented by component 1610 and attributes of data source SRC_DEPT represented by

component 1620 that flow through component 1640 and are thus visible to component 1630. For example, the assignment of the target column TGT_EMPDEPT.NAME can reference SRC_EMP.ENAME without a need to reference to an attribute of component 1640.

[0143] FIG. 17 depicts a flowchart of method 1700 for deriving component attributes based on component type in accordance with an embodiment of the present invention. Implementations of or processing in method 1700 depicted in FIG. 17 may be performed by software (e.g., instructions or code modules) when executed by a central processing unit (CPU or processor) of a logic machine, such as a computer system or information processing device, by hardware components of an electronic device or application-specific integrated circuits, or by combinations of software and hardware elements. Method 1700 depicted in FIG. 17 begins in step 1710.

[0144] In step 1720, a component type for a component is determined. As discussed above, some types of components influence the shape of the data that flows through a mapping while other types of components control the flow of the data but do not fundamentally change the shape of the flow. In step 1730, attributes for downstream components are determined based on the component type. For example, if a component controls the flow of the data but does not fundamentally change the shape of the flow, data integration system 200 may derive a set of attributes based on attributes of upstream components. In step 1740, the derived components are exposed to downstream components. FIG. 17 ends in step 1750. Accordingly, an assignment of target column TGT_EMPDEPT.NAME can reference SRC_EMP.ENAME transparently without a need to reference to an attribute of component 1640.

[0145] Data integration system 200 further makes adding in components to an existing design simple, typically just needing component level connectors to be rerouted. For example, if a filter component were added into a design, changing component level connectors would not require changes to attribute assignments of certain downstream components.

[0146] FIG. 18 is a simplified block diagram of mapping 1600 with filter component 1800 in one embodiment according to the present invention. In this example, mapping 1600 includes filter component 1800 placed between component 1610 and component 1640. To add filter component 1800 (e.g. using filter SRG_EMP.SAL > 3000) into mapping 1600, a user only needs to add filter component 1800 into the graph between component 1610 and

component 1640. The component level connectors would need to be relinked such that filter component 1800 connects to components 1610 as an input and to component 1640 as output. Such as change would not require changes to any downstream assignments in mapping 1600. In traditional flow tools, everything at the column level would need to be relinked by the introduction of the new component.

[0147] FIG. 19 is a simplified block diagram of mapping 1900 in one embodiment according to the present invention. In this example, mapping 1900 includes component S1 representing a data source, component S2 representing a data source, and component T representing a data target. Components C1, C2, and C3 are configured to either influence the shape of the data that flows through a mapping or control the flow of the data but do not fundamentally change the shape of the flow. In this example, component C1 is of a projector type. Therefore, component C1 has a declared set of attributes (some of which are declared from one or more attributes of component S1) as viewed from downstream components (e.g., component C3 et seq.) Component C3 is of a selector type. Therefore, component C3 has a derived set of attributes (some of which are derived from one or more attributes of components C1 and C2) as viewed from downstream components (e.g., component T).

[0148] FIG. 20 depicts a flowchart of method 2000 for generating components in accordance with an embodiment of the present invention. Implementations of or processing in method 2000 depicted in FIG. 20 may be performed by software (e.g., instructions or code modules) when executed by a central processing unit (CPU or processor) of a logic machine, such as a computer system or information processing device, by hardware components of an electronic device or application-specific integrated circuits, or by combinations of software and hardware elements. Method 2000 depicted in FIG. 20 begins in step 2010.

[0149] In step 2020, a component definition is received. For example, a component definition may include rules, operations, procedures, variables, sequences, and the like. In step 2030, a component type is received. For example, if a component changes the shape of data in a flow, the component may be classified in one manner. If the component controls the flow of the data but does not fundamentally change the shape of the flow, the component may be classified in another manner. In step 2040, the component is generated and may be used by data integration system 200. FIG. 20 ends in step 2050.

[0150] Accordingly, data integration system 200 enables users to create a logical design which is platform and technology independent. The user can create a logical design that

defines, at a high level, how a user wants data to flow between sources and targets. The tool can analyze the logical design, in view of the user's infrastructure, and create a physical design. The logical design can include a plurality of components corresponding to each source and target in the design, as well as operations such as joins or filters, and access points. Each component when transferred to the physical design generates code to perform operations on the data. Depending on the underlying technology (e.g., SQL Server, Oracle, Hadoop, etc.) and the language used (SQL, pig, etc.) the code generated by each component may be different.

[0151] Thus, a user of data integration system is not required to specify all data attributes at each component in the logical design, from start to end. Data integration system 200 provides a plurality of component types, such as projector and selector types, that avoid the need to fully declare the information that flows through the logical design. Data integration system 200 is able to decide what attributes are needed at operations represented by predetermined component types. This simplifies both the design and maintenance.

[0152] Conclusion

[0153] FIG. 21 is a simplified block diagram of computer system 2100 that may be used to practice embodiments of the present invention. As shown in FIG. 21, computer system 2100 includes processor 2110 that communicates with a number of peripheral devices via bus subsystem 2120. These peripheral devices may include storage subsystem 2130, comprising memory subsystem 2140 and file storage subsystem 2150, input devices 2160, output devices 2170, and network interface subsystem 2180.

[0154] Bus subsystem 2120 provides a mechanism for letting the various components and subsystems of computer system 2100 communicate with each other as intended. Although bus subsystem 2120 is shown schematically as a single bus, alternative embodiments of the bus subsystem may utilize multiple busses.

[0155] Storage subsystem 2130 may be configured to store the basic programming and data constructs that provide the functionality of the present invention. Software (code modules or instructions) that provides the functionality of the present invention may be stored in storage subsystem 2130. These software modules or instructions may be executed by processor(s) 2110. Storage subsystem 2130 may also provide a repository for storing data used in accordance with the present invention. Storage subsystem 2130 may comprise memory subsystem 2140 and file/disk storage subsystem 2150.

[0156] Memory subsystem 2140 may include a number of memories including a main random access memory (RAM) 2142 for storage of instructions and data during program execution and a read only memory (ROM) 2144 in which fixed instructions are stored. File storage subsystem 2150 provides persistent (non-volatile) storage for program and data files, and may include a hard disk drive, a floppy disk drive along with associated removable media, a Compact Disk Read Only Memory (CD-ROM) drive, a DVD, an optical drive, removable media cartridges, and other like storage media.

[0157] Input devices 2160 may include a keyboard, pointing devices such as a mouse, trackball, touchpad, or graphics tablet, a scanner, a barcode scanner, a touchscreen incorporated into the display, audio input devices such as voice recognition systems, microphones, and other types of input devices. In general, use of the term "input device" is intended to include all possible types of devices and mechanisms for inputting information to computer system 2100.

[0158] Output devices 2170 may include a display subsystem, a printer, a fax machine, or non-visual displays such as audio output devices, etc. The display subsystem may be a cathode ray tube (CRT), a flat-panel device such as a liquid crystal display (LCD), or a projection device. In general, use of the term "output device" is intended to include all possible types of devices and mechanisms for outputting information from computer system 2100.

[0159] Network interface subsystem 2180 provides an interface to other computer systems, devices, and networks, such as communications network 2190. Network interface subsystem 2180 serves as an interface for receiving data from and transmitting data to other systems from computer system 2100. Some examples of communications network 2190 are private networks, public networks, leased lines, the Internet, Ethernet networks, token ring networks, fiber optic networks, and the like.

[0160] Computer system 2100 can be of various types including a personal computer, a portable computer, a workstation, a network computer, a mainframe, a kiosk, or any other data processing system. Due to the ever-changing nature of computers and networks, the description of computer system 2100 depicted in FIG. 21 is intended only as a specific example for purposes of illustrating the preferred embodiment of the computer system. Many other configurations having more or fewer components than the system depicted in FIG. 21 are possible.

[0161] FIG. 22 is a simplified block diagram of system 2200 for facilitating generation of a data mapping according to an embodiment. The units included in system 2200 in FIG. 22 can be implemented in software (e.g., instructions or code modules) which can be executed by a central processing unit (CPU or processor) of a logic machine, such as a computer system or information processing device, in hardware components of an electronic device or application-specific integrated circuits, or in combinations of software and hardware elements.

[0162] As shown in FIG. 22, among others, data integration system 2200 may include receiving unit 2210, determining unit 2220 and generating unit 2230. Receiving unit 2210 may be configured to receive information specifying one or more components of a logical design, wherein at least one of the one or more components is of a first type. Determining unit 2220 may be configured to determine a set of data attributes visible to downstream components in the logical design of the at least one of the one or more components that is of the first type based on upstream components in the logical design. Generating unit 2230 may be configured to generate information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type.

[0163] According to an embodiment, determining the set of data attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type may comprise deriving one or more attributes visible from an upstream component and exposing the one or more attributes to a downstream component.

[0164] According to an embodiment, the information specifying the one or more components of the logical design may comprise information indicative of an operation that changes shape of the information flowing through the logical design.

[0165] According to an embodiment, the information specifying the one or more components of the logical design may comprise information indicative of an operation that controls the flow of information flowing through the logical design but does not change shape of the information flowing through the logical design.

[0166] According to an embodiment, the information specifying the one or more components of the logical design may comprise information indicative of a source component having one or more attributes of data stored in a source datastore.

[0167] According to an embodiment, the information specifying the one or more components of the logical design may comprise information indicative of a target component having one or more attributes of data to be stored in a target datastore.

[0168] According to an embodiment, generating the information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type may comprise exporting a list of attributes to a downstream component.

[0169] According to an embodiment, receiving unit 2210 may be further configured to receive a change in the logical design through the introduction or removal of a component or an attribute into the logical design. Determining unit 2220 may be further configured to determine whether the change in the logical design affects the at least one of the one or more components that is of the first type. Based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, determining unit 2220 may be further configured to determine an updated set of data attributes visible to downstream components.

[0170] According to an embodiment, data integration system 2200 may further include preserving unit 2240. Receiving unit 2210 may be further configured to receive a change in the logical design through the introduction of a component or an attribute into the logical design. Determining unit 2220 may be further configured to determine whether the change in the logical design affects the at least one of the one or more components that is of the first type. Based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, preserving unit 2240 may be configured to preserve the set of data attributes visible to downstream components.

[0171] According to an embodiment, data integration system 2200 may further include renaming unit 2250. Receiving unit 2210 may be further configured to receive a change in the logical design renaming a component or an attribute. Determining unit 2220 may be further configured to determine whether the change in the logical design affects the at least one of the one or more components that is of the first type. Based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, renaming unit 2250 may be configured to rename the set of data attributes visible to downstream components.

[0172] **[0173]** Although specific embodiments of the invention have been described, various modifications, alterations, alternative constructions, and equivalents are also encompassed within the scope of the invention. The described invention is not restricted to operation within certain specific data processing environments, but is free to operate within a plurality of data processing environments. Additionally, although the present invention has been described using a particular series of transactions and steps, it should be apparent to those skilled in the art that the scope of the present invention is not limited to the described series of transactions and steps.

[0174] Further, while the present invention has been described using a particular combination of hardware and software, it should be recognized that other combinations of hardware and software are also within the scope of the present invention. The present invention may be implemented only in hardware, or only in software, or using combinations thereof.

[0175] The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense. It will, however, be evident that additions, subtractions, deletions, and other modifications and changes may be made thereunto without departing from the broader spirit and scope of the invention as set forth in the claims.

[0176] Various embodiments of any of one or more inventions whose teachings may be presented within this disclosure can be implemented in the form of logic in software, firmware, hardware, or a combination thereof. The logic may be stored in or on a machine-accessible memory, a machine-readable article, a tangible computer-readable medium, a computer-readable storage medium, or other computer/machine-readable media as a set of instructions adapted to direct a central processing unit (CPU or processor) of a logic machine to perform a set of steps that may be disclosed in various embodiments of an invention presented within this disclosure. The logic may form part of a software program or computer program product as code modules become operational with a processor of a computer system or an information-processing device when executed to perform a method or process in various embodiments of an invention presented within this disclosure. Based on this disclosure and the teachings provided herein, a person of ordinary skill in the art will appreciate other ways, variations, modifications, alternatives, and/or methods for implementing in software, firmware, hardware, or combinations thereof any of the disclosed

operations or functionalities of various embodiments of one or more of the presented inventions.

[0177] The disclosed examples, implementations, and various embodiments of any one of those inventions whose teachings may be presented within this disclosure are merely illustrative to convey with reasonable clarity to those skilled in the art the teachings of this disclosure. As these implementations and embodiments may be described with reference to exemplary illustrations or specific figures, various modifications or adaptations of the methods and/or specific structures described can become apparent to those skilled in the art. All such modifications, adaptations, or variations that rely upon this disclosure and these teachings found herein, and through which the teachings have advanced the art, are to be considered within the scope of the one or more inventions whose teachings may be presented within this disclosure. Hence, the present descriptions and drawings should not be considered in a limiting sense, as it is understood that an invention presented within a disclosure is in no way limited to those embodiments specifically illustrated.

[0178] Accordingly, the above description and any accompanying drawings, illustrations, and figures are intended to be illustrative but not restrictive. The scope of any invention presented within this disclosure should, therefore, be determined not with simple reference to the above description and those embodiments shown in the figures, but instead should be determined with reference to the pending claims along with their full scope or equivalents.

WHAT IS CLAIMED IS:

1. A method for facilitating generation of a data mapping, the method comprising:

receiving, at one or more computer systems, information specifying one or more components of a logical design, wherein at least one of the one or more components is of a first type;

determining, with one or more processors associated with the one or more computer systems, a set of data attributes visible to downstream components in the logical design of the at least one of the one or more components that is of the first type based on upstream components in the logical design; and

generating, with the one or more processors associated with the one or more computer systems, information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type.

2. The method of claim 1 wherein determining, with the one or more processors associated with the one or more computer systems, the set of data attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type comprises deriving one or more attributes visible from an upstream component and exposing the one or more attributes to a downstream component.

3. The method of claim 1 or 2 wherein receiving the information specifying the one or more components of the logical design comprises receiving information indicative of an operation that changes shape of the information flowing through the logical design.

4. The method of any one of claims 1 to 3 wherein receiving the information specifying the one or more components of the logical design comprises receiving information indicative of an operation that controls the flow of information flowing through the logical design but does not change shape of the information flowing through the logical design.

5. The method of any one of claims 1 to 4 wherein receiving the information specifying the one or more components of the logical design comprises receiving information

indicative of a source component having one or more attributes of data stored in a source datastore.

6. The method of any one of claims 1 to 5 wherein receiving the information specifying the one or more components of the logical design comprises receiving information indicative of a target component having one or more attributes of data to be stored in a target datastore.

7. The method of any one of claims 1 to 6 wherein generating the information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type comprises exporting a list of attributes to a downstream component.

8. The method of any one of claims 1 to 7 further comprising:

receiving, at the one or more computer systems, a change in the logical design through the introduction or removal of a component or an attribute into the logical design;

determining, with the one or more processors associated with the one or more computer systems, whether the change in the logical design affects the at least one of the one or more components that is of the first type; and

based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, determining, with the one or more processors associated with the one or more computer systems, an updated set of data attributes visible to downstream components.

9. The method of any one of claims 1 to 8 further comprising:

receiving, at the one or more computer systems, a change in the logical design through the introduction of a component or an attribute into the logical design;

determining, with the one or more processors associated with the one or more computer systems, whether the change in the logical design affects the at least one of the one or more components that is of the first type; and

based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, preserving the set of data attributes visible to downstream components.

10. The method of any one of claims 1 to 9 further comprising:

receiving, at the one or more computer systems, a change in the logical design renaming a component or an attribute;

determining, with the one or more processors associated with the one or more computer systems, whether the change in the logical design affects the at least one of the one or more components that is of the first type; and

based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, renaming the set of data attributes visible to downstream components.

11. A non-transitory computer-readable medium storing computer-executable code for facilitating generation of a data mapping, the non-transitory computer-readable medium comprising:

code for receiving information specifying one or more components of a logical design, wherein at least one of the one or more components is of a first type;

code for determining a set of data attributes visible to downstream components in the logical design of the at least one of the one or more components that is of the first type based on upstream components in the logical design; and

code for generating information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type.

12. The non-transitory computer-readable medium of claim 11 wherein the code for determining the set of data attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type comprises code for deriving one or more attributes visible from an upstream component and exposing the one or more attributes to a downstream component.

13. The non-transitory computer-readable medium of any one of claim 11 or 12 wherein the code for receiving the information specifying the one or more components of the logical design comprises code for receiving information indicative of an operation that changes shape of the information flowing through the logical design.

14. The non-transitory computer-readable medium of any one of claims 11 to 13 wherein the code for receiving the information specifying the one or more components of the logical design comprises code for receiving information indicative of an operation that controls the flow of information flowing through the logical design but does not change shape of the information flowing through the logical design.

15. The non-transitory computer-readable medium of any one of claims 11 to 14 wherein the code for receiving the information specifying the one or more components of the logical design comprises code for receiving information indicative of a source component having one or more attributes of data stored in a source datastore.

16. The non-transitory computer-readable medium of any one of claims 11 to 15 wherein the code for receiving the information specifying the one or more components of the logical design comprises code for receiving information indicative of a target component having one or more attributes of data to be stored in a target datastore.

17. The non-transitory computer-readable medium of any one of claims 11 to 16 wherein the code for generating the information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type comprises code for exporting a list of attributes to a downstream component.

18. The non-transitory computer-readable medium of any one of claims 11 to 17 further comprising:

code for receiving a change in the logical design through the introduction or removal of a component or an attribute into the logical design;

code for determining whether the change in the logical design affects the at least one of the one or more components that is of the first type; and

based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, code for determining an updated set of data attributes visible to downstream components.

19. The non-transitory computer-readable medium of any one of claims 11 to 18 further comprising:

code for receiving a change in the logical design through the introduction or removal of a component or an attribute into the logical design;

code for determining whether the change in the logical design affects the at least one of the one or more components that is of the first type; and

based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, code for preserving the set of data attributes visible to downstream components.

20. The non-transitory computer-readable medium of any one of claims 11 to 19 further comprising:

code for receiving a change in the logical design renaming a component or an attribute;

code for determining whether the change in the logical design affects the at least one of the one or more components that is of the first type; and

based on a determination that the change in the logical design affects the at least one of the one or more components that is of the first type, code for renaming the set of data attributes visible to downstream components.

21. A system for facilitating generation of a data mapping, the system comprising:

a processor; and

a memory in communication with the processor and configured to store a set of instructions which when executed by the processor configure the processor to:

receive information specifying one or more components of a logical design, wherein at least one of the one or more components is of a first type;

determine a set of data attributes visible to downstream components in the logical design of the at least one of the one or more components that is of the first type based on upstream components in the logical design; and

generate information indicative of the set of attributes visible to the downstream components in the logical design of the at least one of the one or more components that is of the first type.

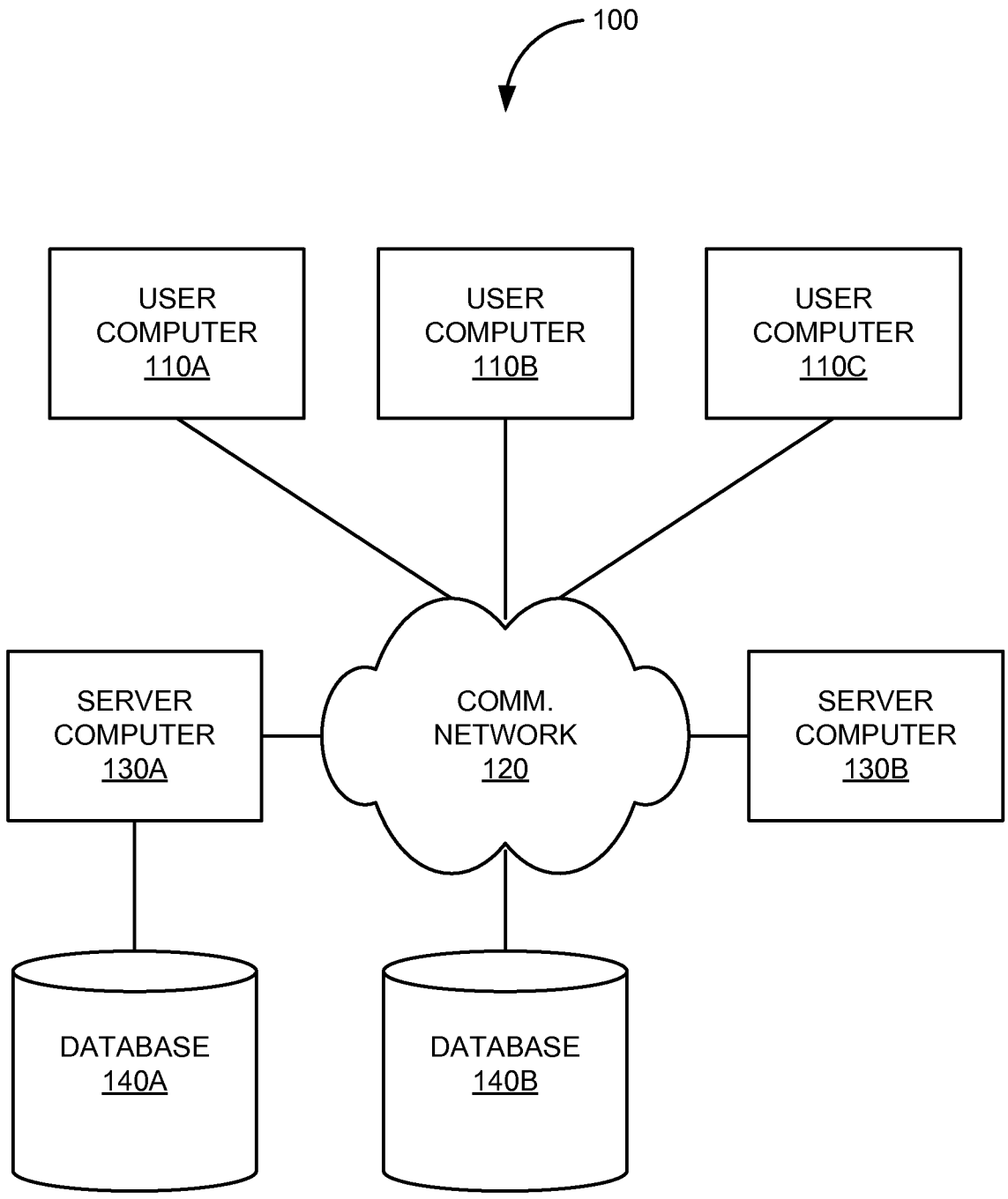
100
USER COMPUTER 110A
USER COMPUTER 110B
USER COMPUTER 110C
SERVER COMPUTER 130A
COMM. NETWORK 120
SERVER COMPUTER 130B
DATABASE 140A
DATABASE 140B

FIG. 1

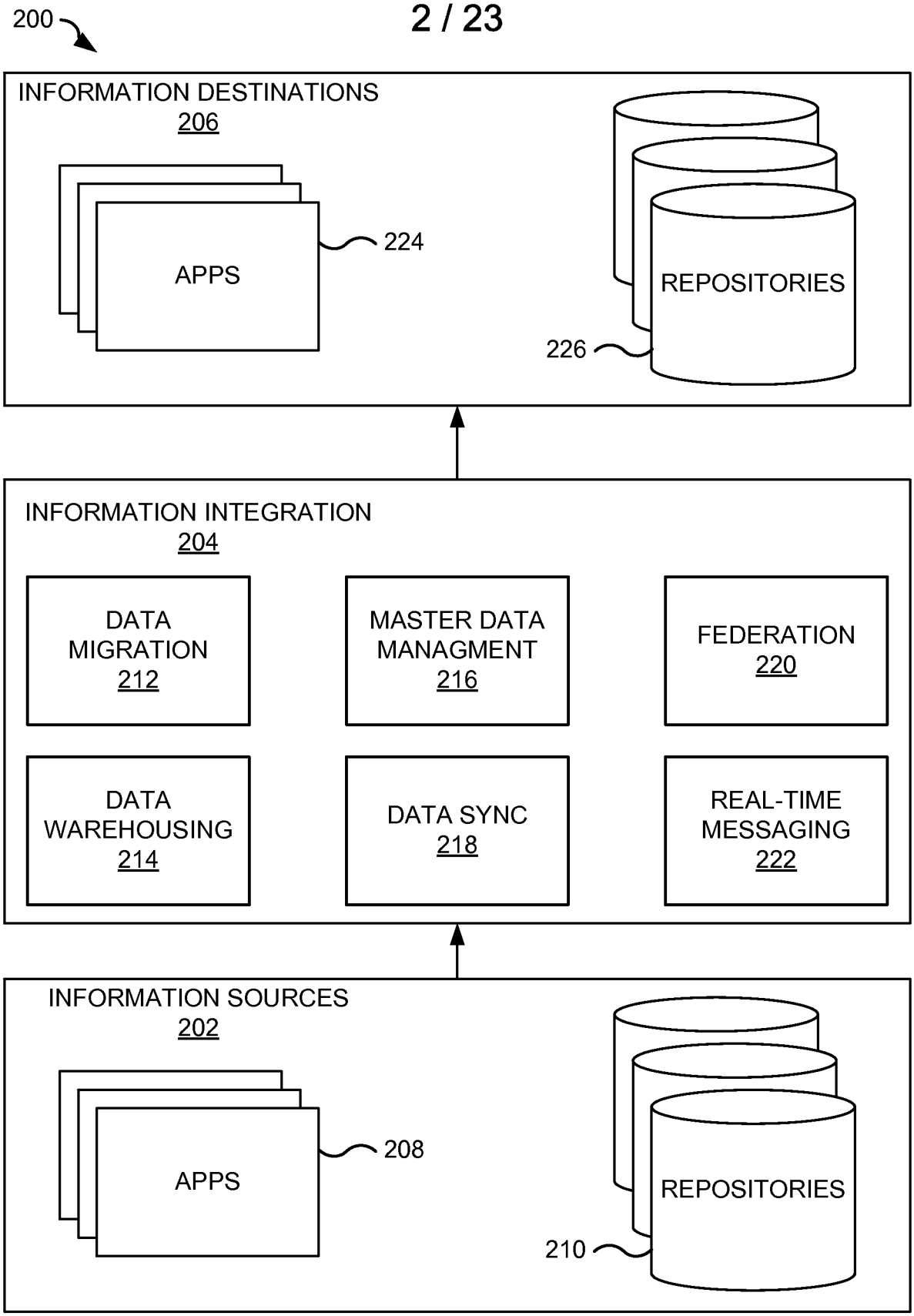
200
INFORMATION DESTINATIONS
206
APPS
224
REPOSITORIES
226
INFORMATION INTEGRATION
204
DATA MIGRATION
212
MASTER DATA MANAGMENT
216
FEDERATION
220
DATA WAREHOUSING
214
DATA SYNC
218
REAL-TIME MESSAGING
222
INFORMATION SOURCES
202
APPS
208
REPOSITORIES
210

FIG. 2

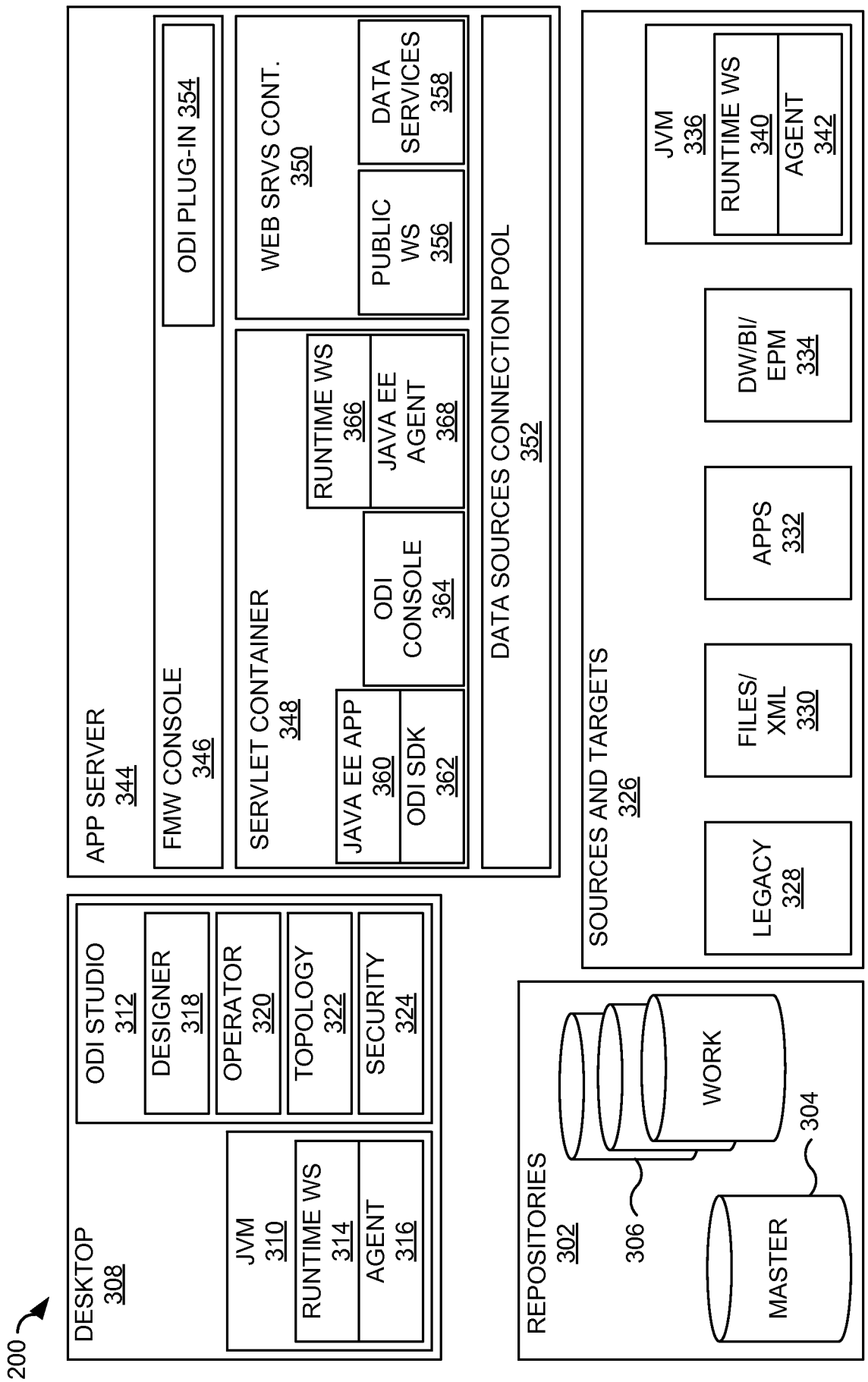
200
DESKTOP
308
ODI STUDIO
312
DESIGNER
318
OPERATOR
320
TOPOLOGY
322
SECURITY
324
JVM
310
RUNTIME WS
314
AGENT
316
APP SERVER
344
FMW CONSOLE
346
ODI PLUG-IN 354
SERVLET CONTAINER
348
JAVA EE APP
360
ODI SDK
362
ODI
CONSOLE
364
RUNTIME WS
366
JAVA EE
AGENT
368
WEB SRVS CONT.
350
PUBLIC
WS
356
DATA
SERVICES
358
DATA SOURCES CONNECTION POOL
352
REPOSITORIES
302
306
WORK
MASTER
304
SOURCES AND TARGETS
326
LEGACY
328
FILES/
XML
330
APPS
332
DW/BI/
EPM
334
JVM
336
RUNTIME WS
340
AGENT
342

FIG. 3

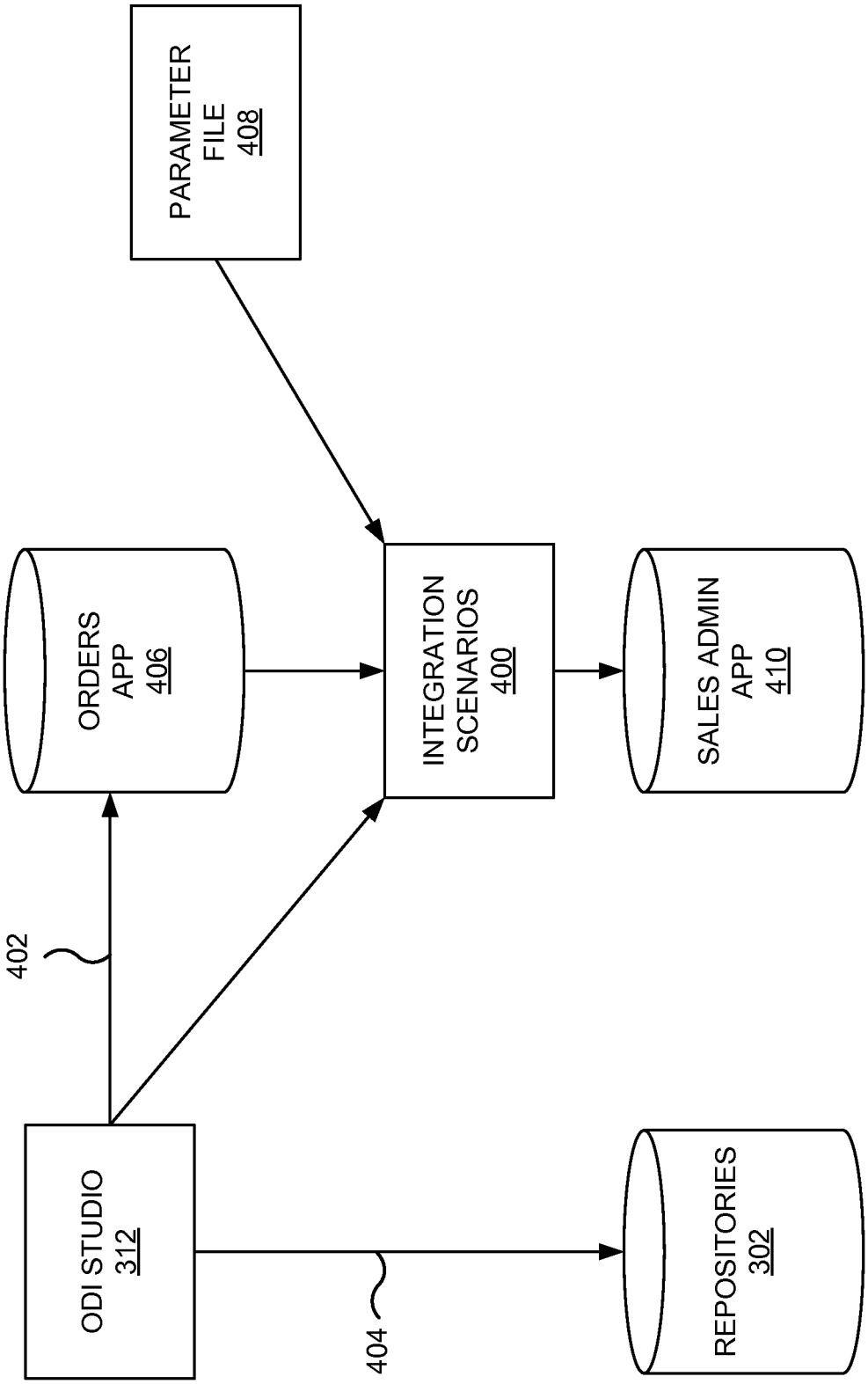
PARAMETER FILE 408
ORDERS APP 406
INTEGRATION SCENARIOS 400
SALES ADMIN APP 410
402
ODI STUDIO 312
404
REPOSITORIES 302

FIG. 4

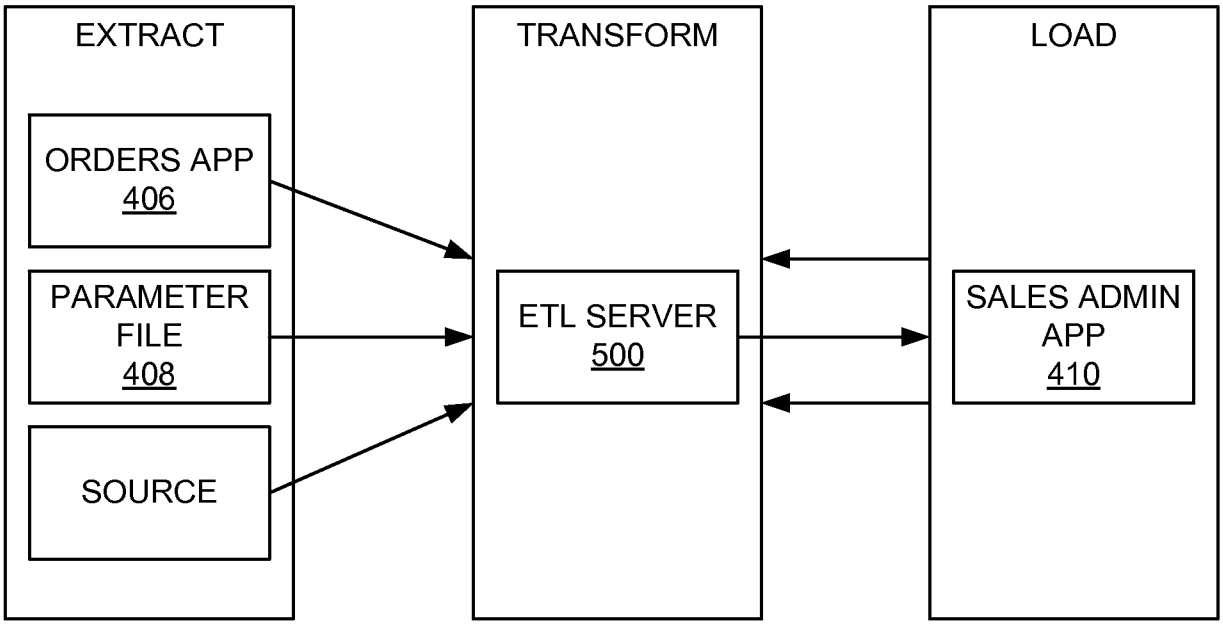
EXTRACT
ORDERS APP
406
PARAMETER
FILE
408
SOURCE
TRANSFORM
ETL SERVER
500
LOAD
SALES ADMIN
APP
410

FIG. 5A

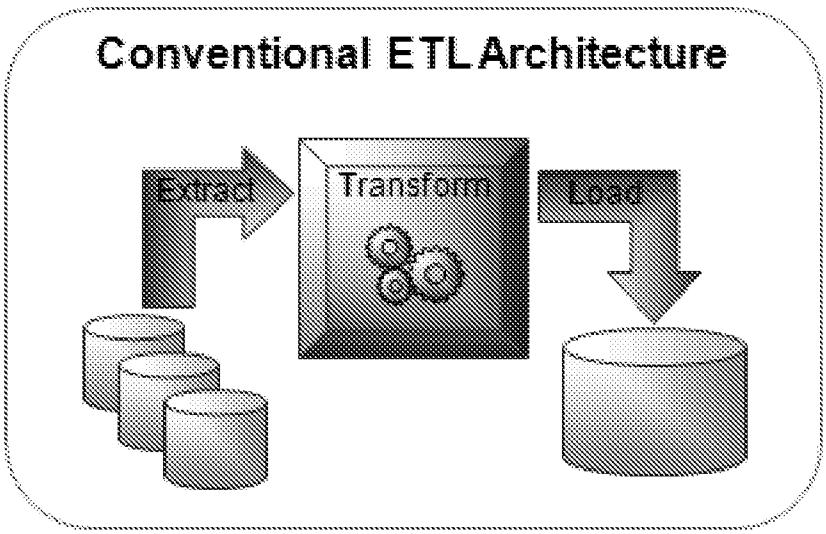
Conventional ETL Architecture
Extract
Transform
Load

FIG. 5B

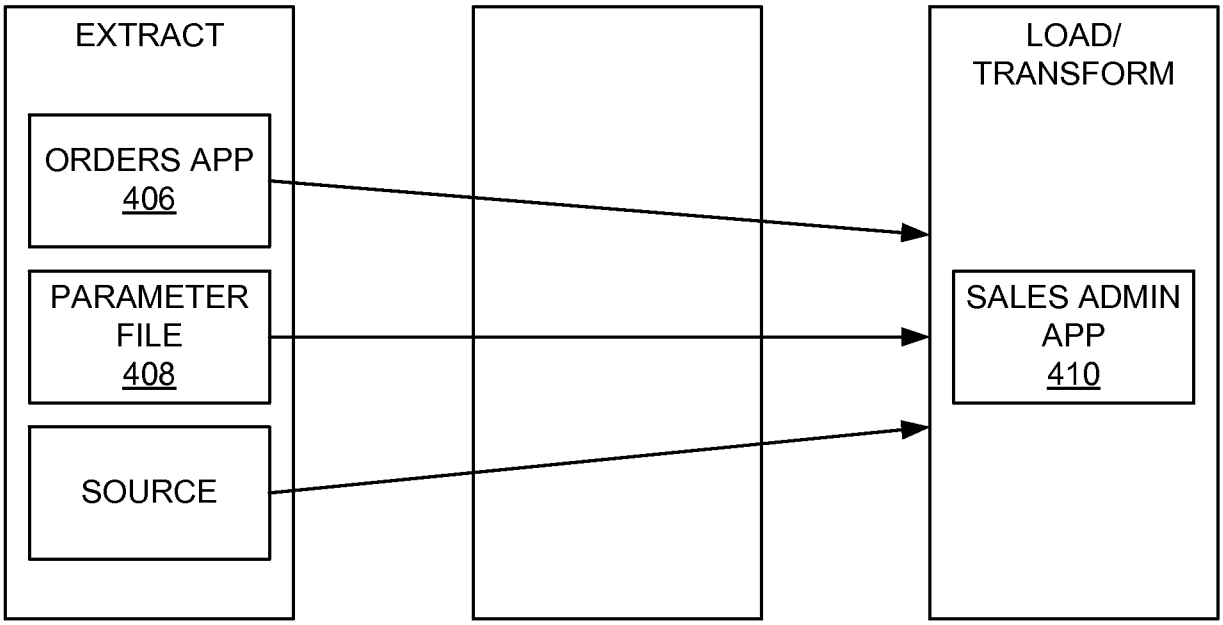
EXTRACT
ORDERS APP
406
PARAMETER
FILE
408
SOURCE
LOAD/
TRANSFORM
SALES ADMIN
APP
410

FIG. 6A

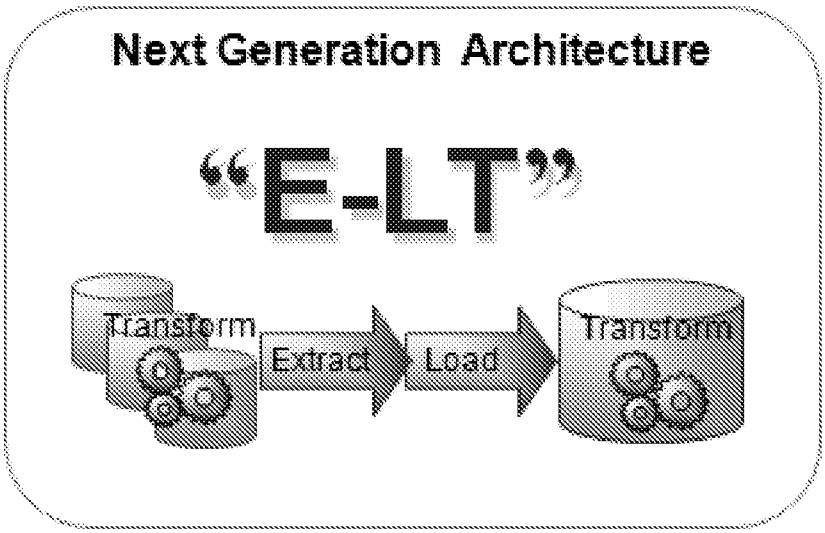
Next Generation Architecture
"E-LT"
Transform
Extract
Load
Transform

FIG. 6B

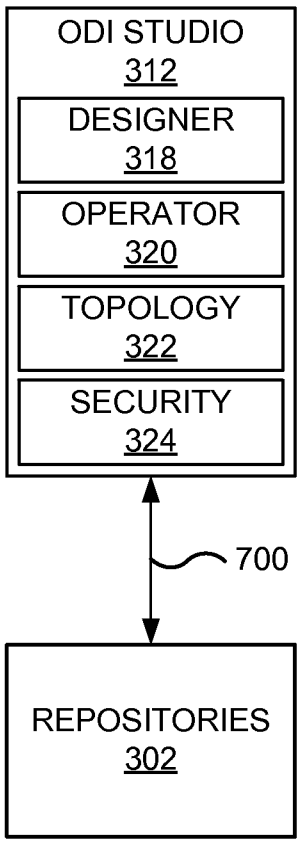
ODI STUDIO
312
DESIGNER
318
OPERATOR
320
TOPOLOGY
322
SECURITY
324
700
REPOSITORIES
302

FIG. 7

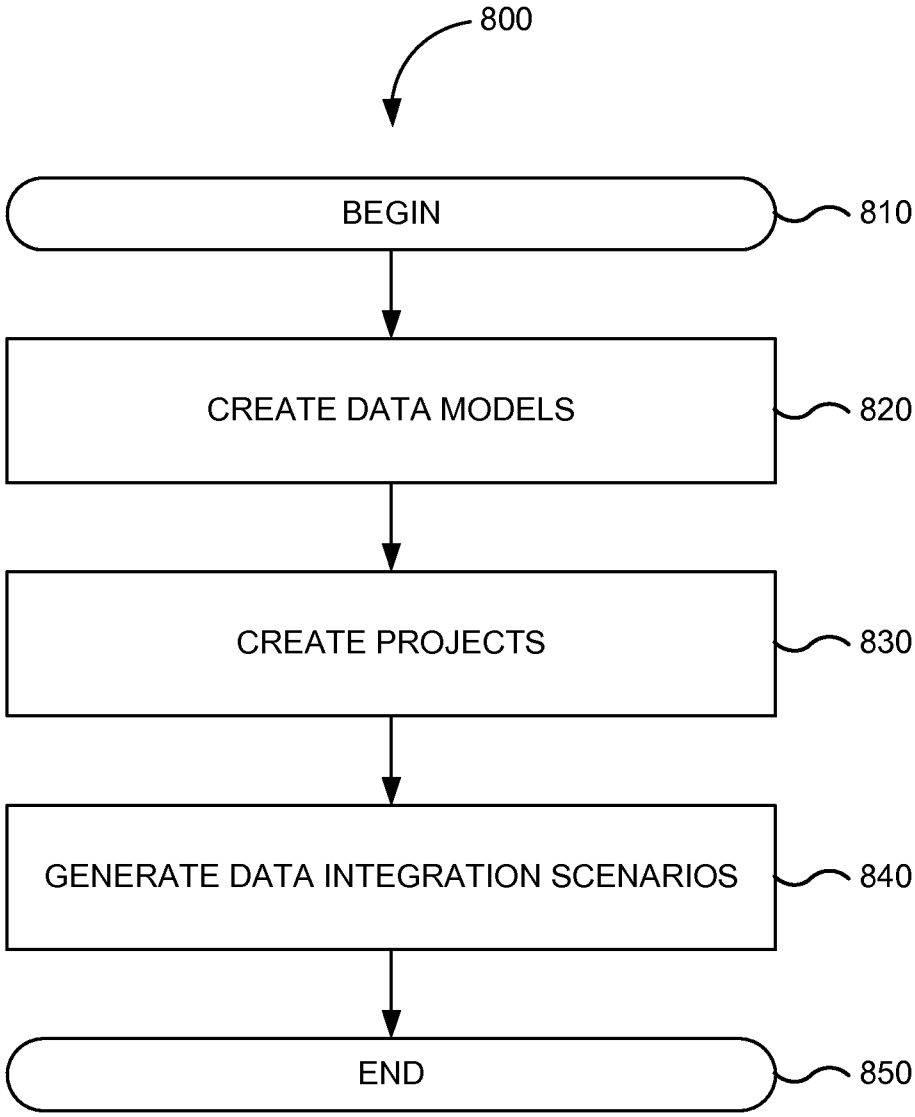
800
BEGIN
810
CREATE DATA MODELS
820
CREATE PROJECTS
830
GENERATE DATA INTEGRATION SCENARIOS
840
END
850

FIG. 8

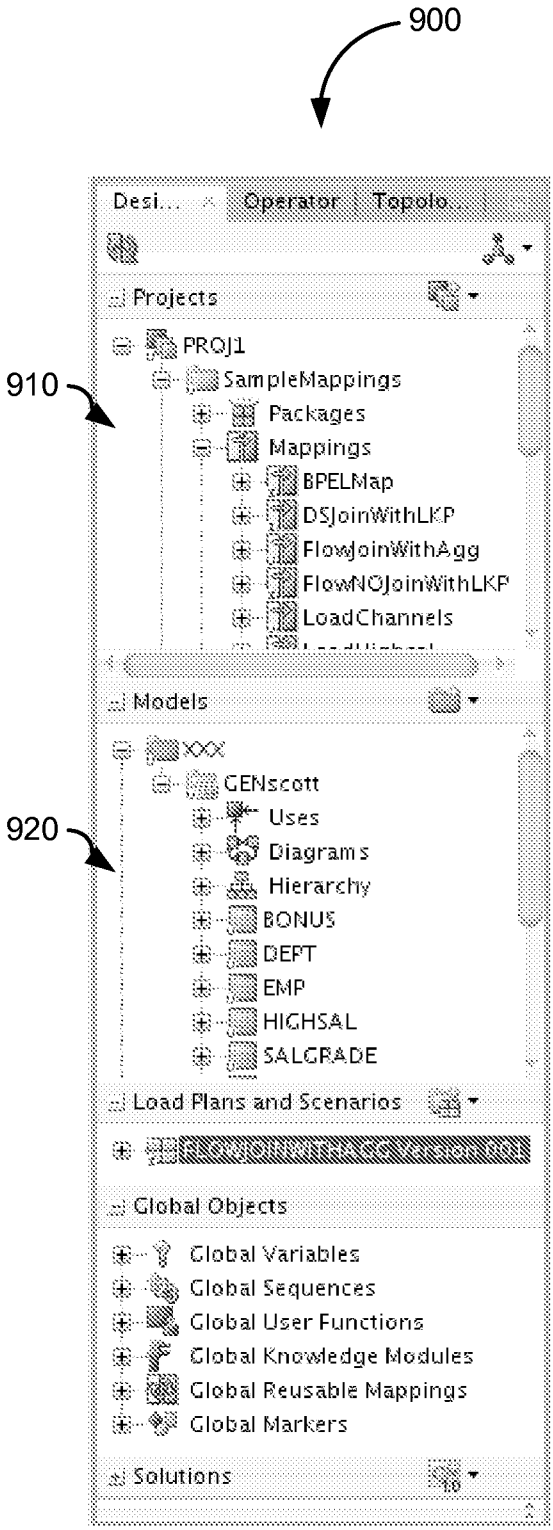
900
910
920
Desi...
Operator
Topolo...
Projects
PROJ1
SampleMappings
Packages
Mappings
BPELMap
DSJoinWithLKP
FlowJoinWithAgg
FlowNOJoinWithLKP
LoadChannels
Models
XXX
GENscott
Uses
Diagrams
Hierarchy
BONUS
DEPT
EMP
HIGHSAL
SALGRADE
Load Plans and Scenarios
Global Objects
Global Variables
Global Sequences
Global User Functions
Global Knowledge Modules
Global Reusable Mappings
Global Markers
Solutions

FIG. 9

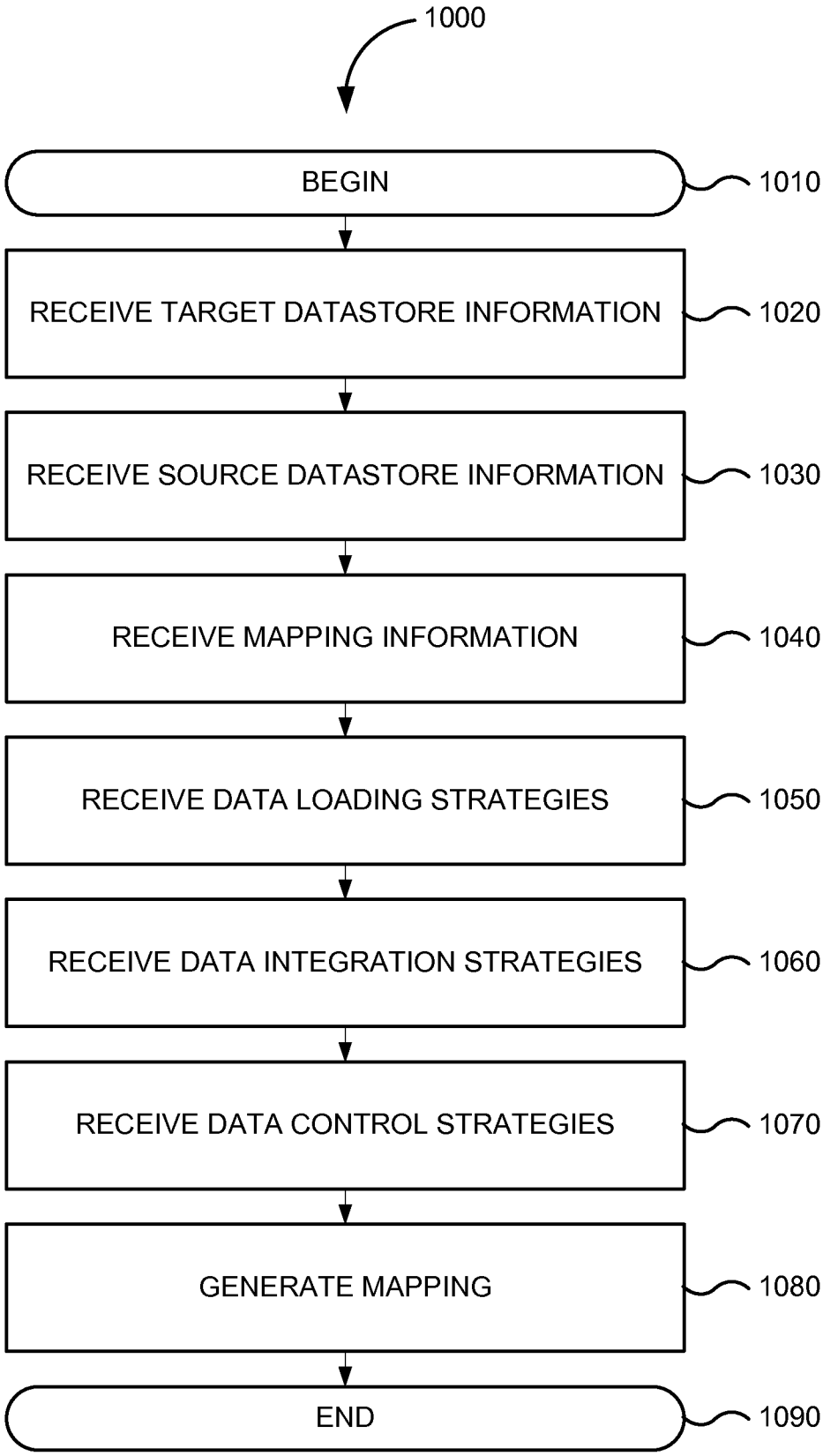
1000
BEGIN
1010
RECEIVE TARGET DATASTORE INFORMATION
1020
RECEIVE SOURCE DATASTORE INFORMATION
1030
RECEIVE MAPPING INFORMATION
1040
RECEIVE DATA LOADING STRATEGIES
1050
RECEIVE DATA INTEGRATION STRATEGIES
1060
RECEIVE DATA CONTROL STRATEGIES
1070
GENERATE MAPPING
1080
END
1090

FIG. 10

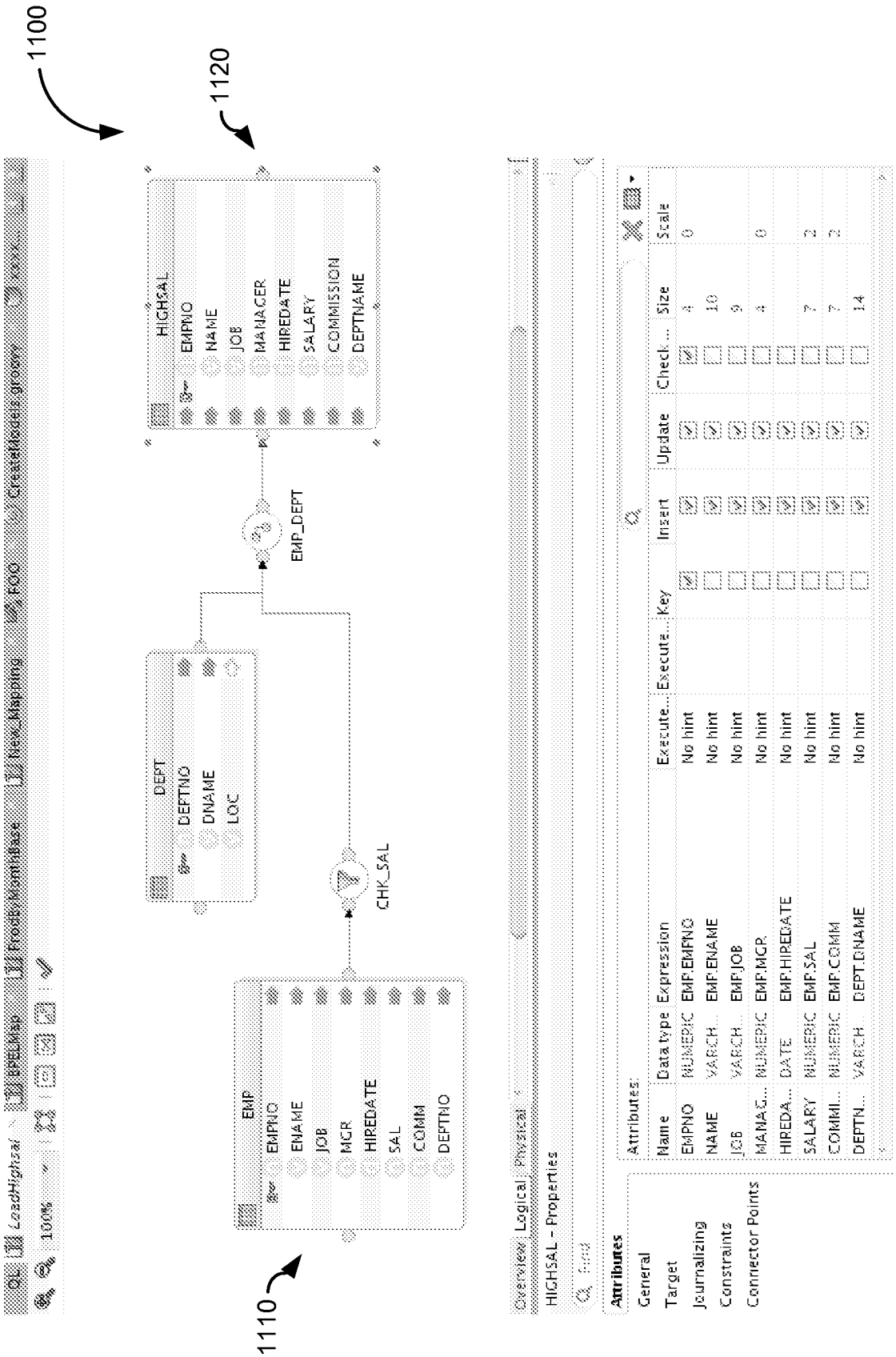
1100
1120
1110
HIGHSAL
EMPNO
NAME
JOB
MANAGER
HIREDATE
SALARY
COMMISSION
DEPTNAME
DEPT
DEPTNO
DNAME
LOC
EMP
EMPNO
ENAME
JOB
MGR
HIREDATE
SAL
COMM
DEPTNO
EMP_DEPT
CHK_SAL
Overview
Logical
Physical
HIGHSAL - Properties
Attributes
General
Target
Journalizing
Constraints
Connector Points
Attributes
Name
Datatype
Expression
Execute...
Execute...
Key
Insert
Update
Check ...
Size
Scale
EMPNO
NUMERIC
EMP.EMPNO
No hint
NAME
VARCH...
EMP.ENAME
No hint
JOB
VARCH...
EMP.JOB
No hint
MANAG...
NUMERIC
EMP.MGR
No hint
HIREDA...
DATE
EMP.HIREDATE
No hint
SALARY
NUMERIC
EMP.SAL
No hint
COMMI...
NUMERIC
EMP.COMM
No hint
DEPTN...
VARCH...
DEPT.DNAME
No hint

FIG. 11

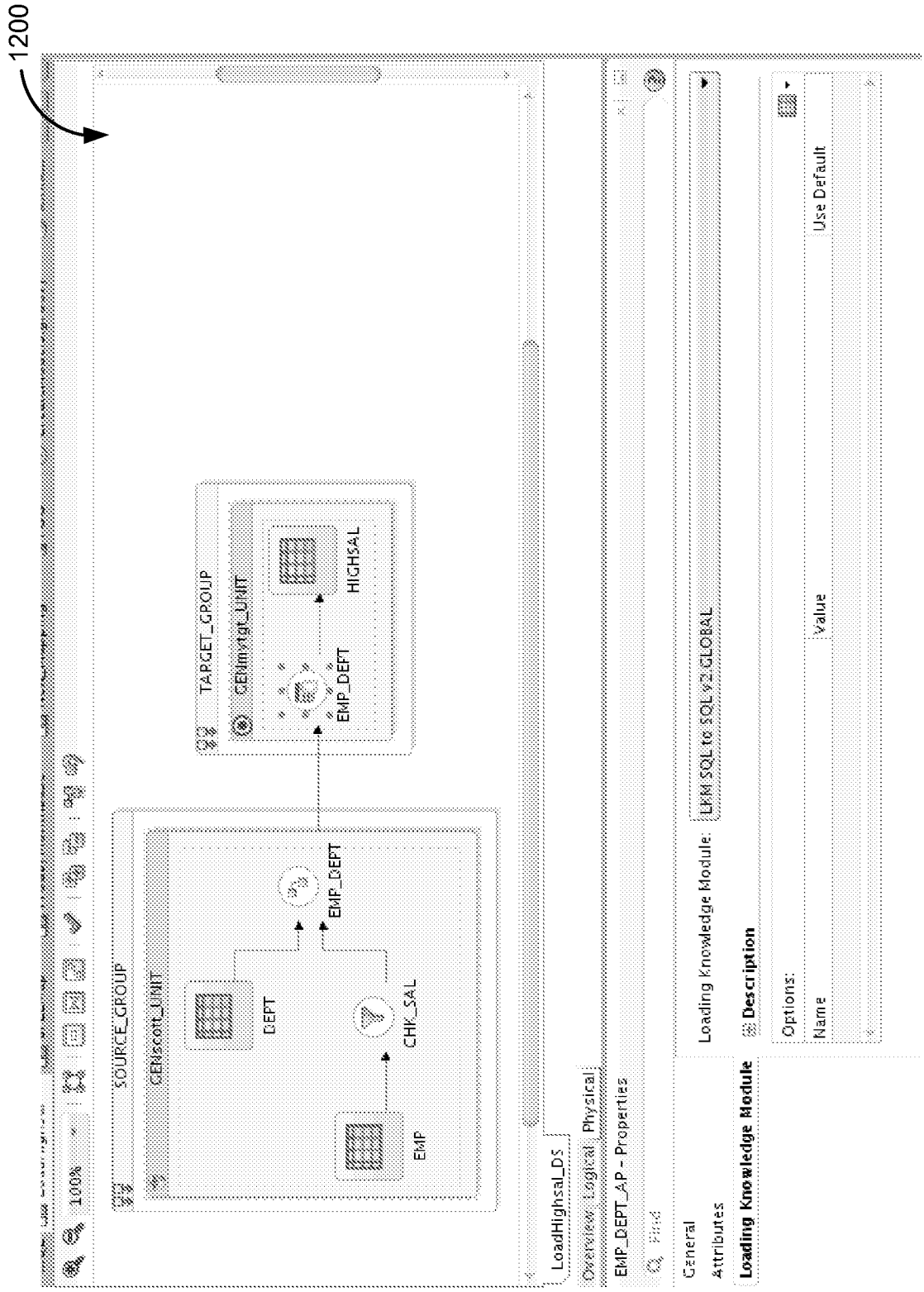
1200
SOURCE_GROUP
GEMscott_UNIT
DEPT
EMP_DEPT
CHK_SAL
EMP
TARGET_GROUP
GEMtgt_UNIT
EMP_DEPT
HIGHSAL
LoadHighsal_DS
Overview Logical Physical
EMP_DEPT_AP - Properties
General
Attributes
Loading Knowledge Module
Loading Knowledge Module:
LKM SQL to SQL v2 GLOBAL
Description
Options:
Name
Value
Use Default

FIG. 12

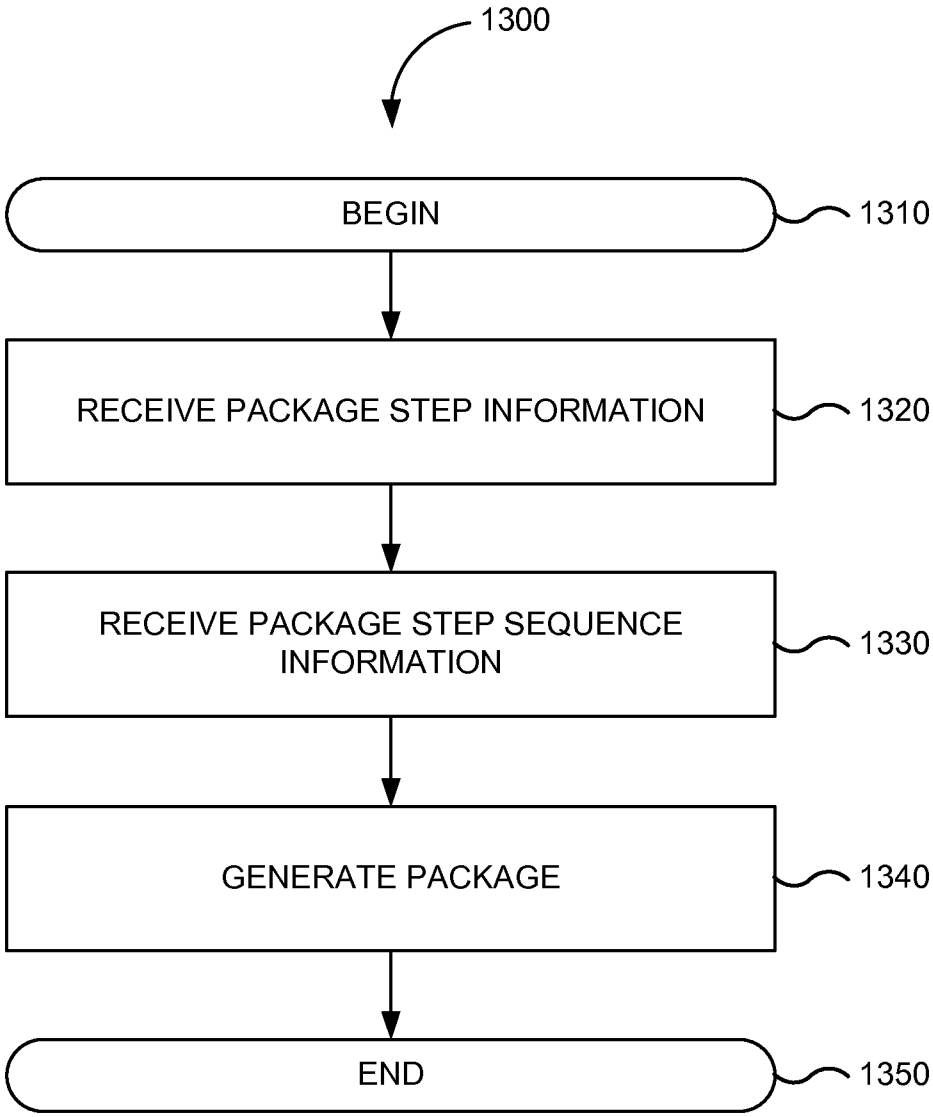
1300
BEGIN
1310
RECEIVE PACKAGE STEP INFORMATION
1320
RECEIVE PACKAGE STEP SEQUENCE INFORMATION
1330
GENERATE PACKAGE
1340
END
1350

FIG. 13

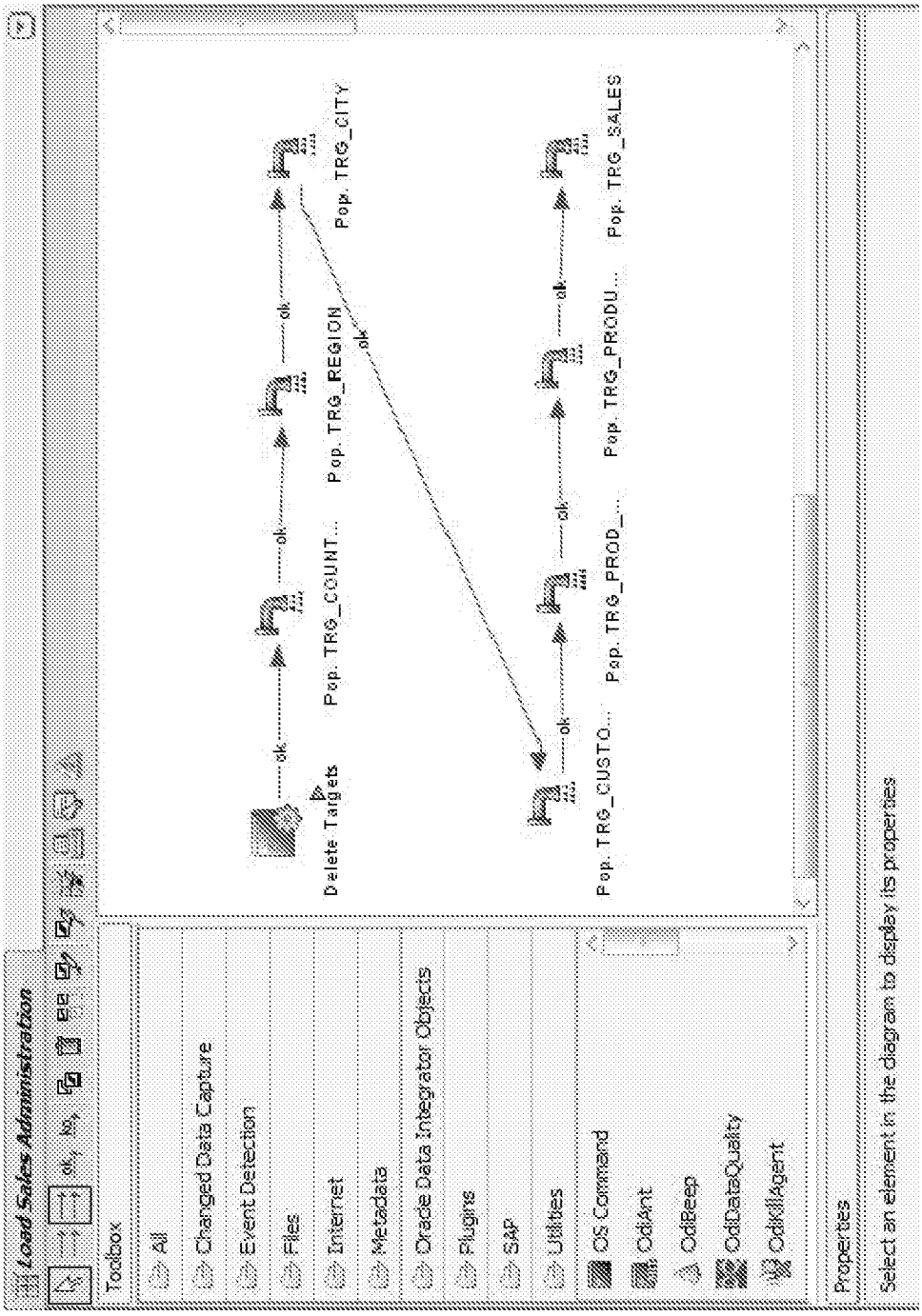
Load Sales Administration
Toolbox
All
Changed Data Capture
Event Detection
Files
Internet
Metadata
Oracle Data Integrator Objects
Plugins
SAP
Utilities
OS Command
OdiAnt
OdiBeep
OdiDataQuality
OdiXmlAgent
Delete Targets
Pop. TRG_COUNT...
Pop. TRG_REGION
Pop. TRG_CITY
Pop. TRG_CUSTO...
Pop. TRG_PROD_...
Pop. TRG_PRODU...
Pop. TRG_SALES
ok
Properties
Select an element in the diagram to display its properties

FIG. 14

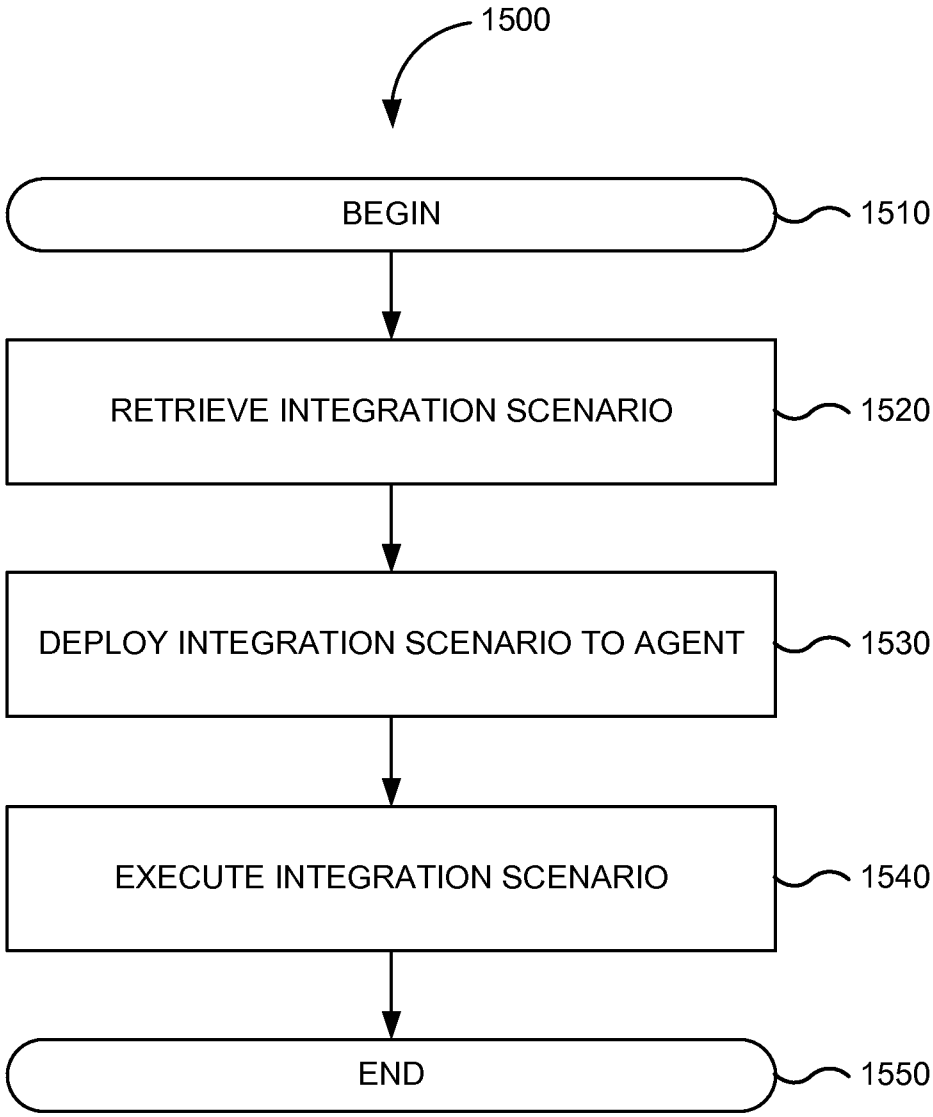
1500
BEGIN
1510
RETRIEVE INTEGRATION SCENARIO
1520
DEPLOY INTEGRATION SCENARIO TO AGENT
1530
EXECUTE INTEGRATION SCENARIO
1540
END
1550

FIG. 15

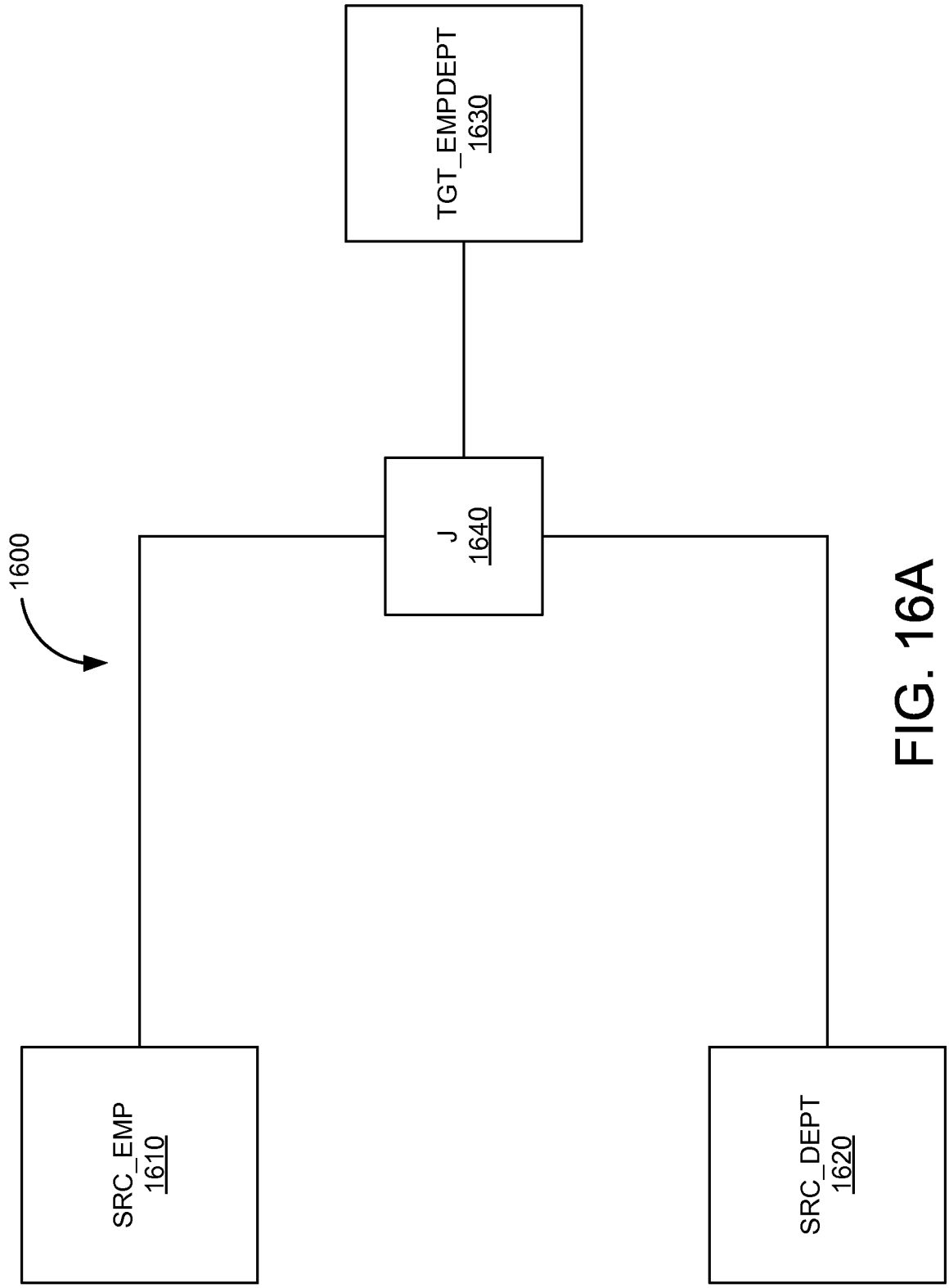

FIG. 16A

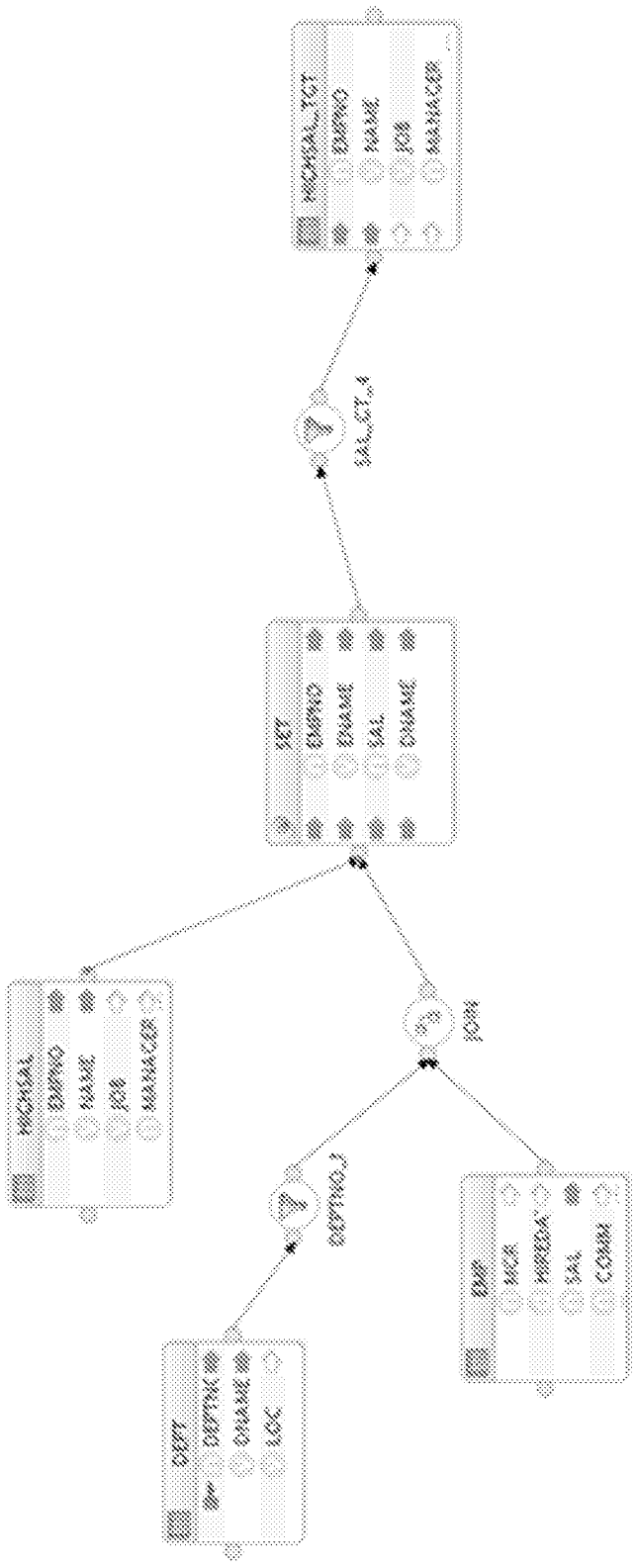
REGIONAL_TGT
DEPNO
NAME
JOB
MANAGER
SAL_GT_4
EMP
DEPNO
ENAME
SAL
ONAME
JOIN
REGIONAL
DEPNO
NAME
JOB
MANAGER
DEPNO_J
DEPT
DEPTNO
DNAME
LOC
EMP
MGR
HIREDATE
SAL
COMM

FIG. 16B

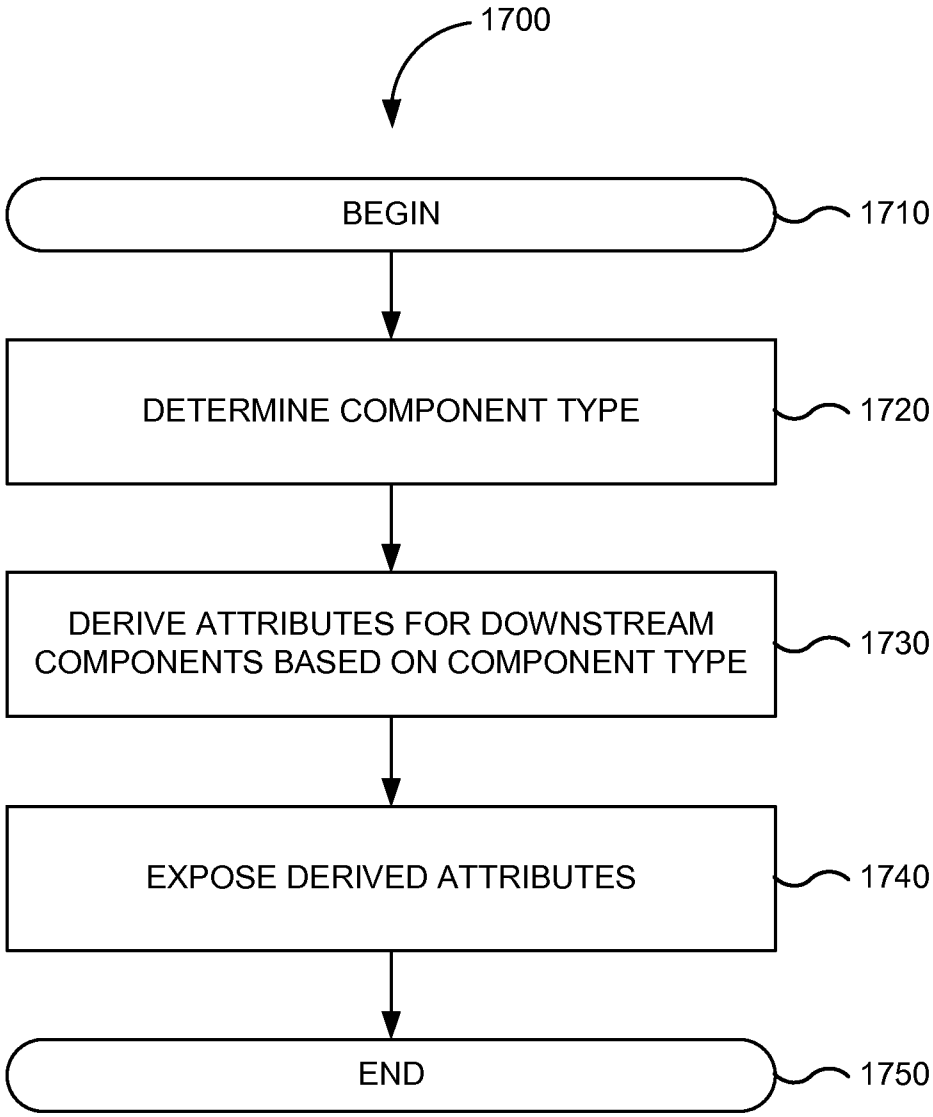
1700
BEGIN
1710
DETERMINE COMPONENT TYPE
1720
DERIVE ATTRIBUTES FOR DOWNSTREAM COMPONENTS BASED ON COMPONENT TYPE
1730
EXPOSE DERIVED ATTRIBUTES
1740
END
1750

FIG. 17

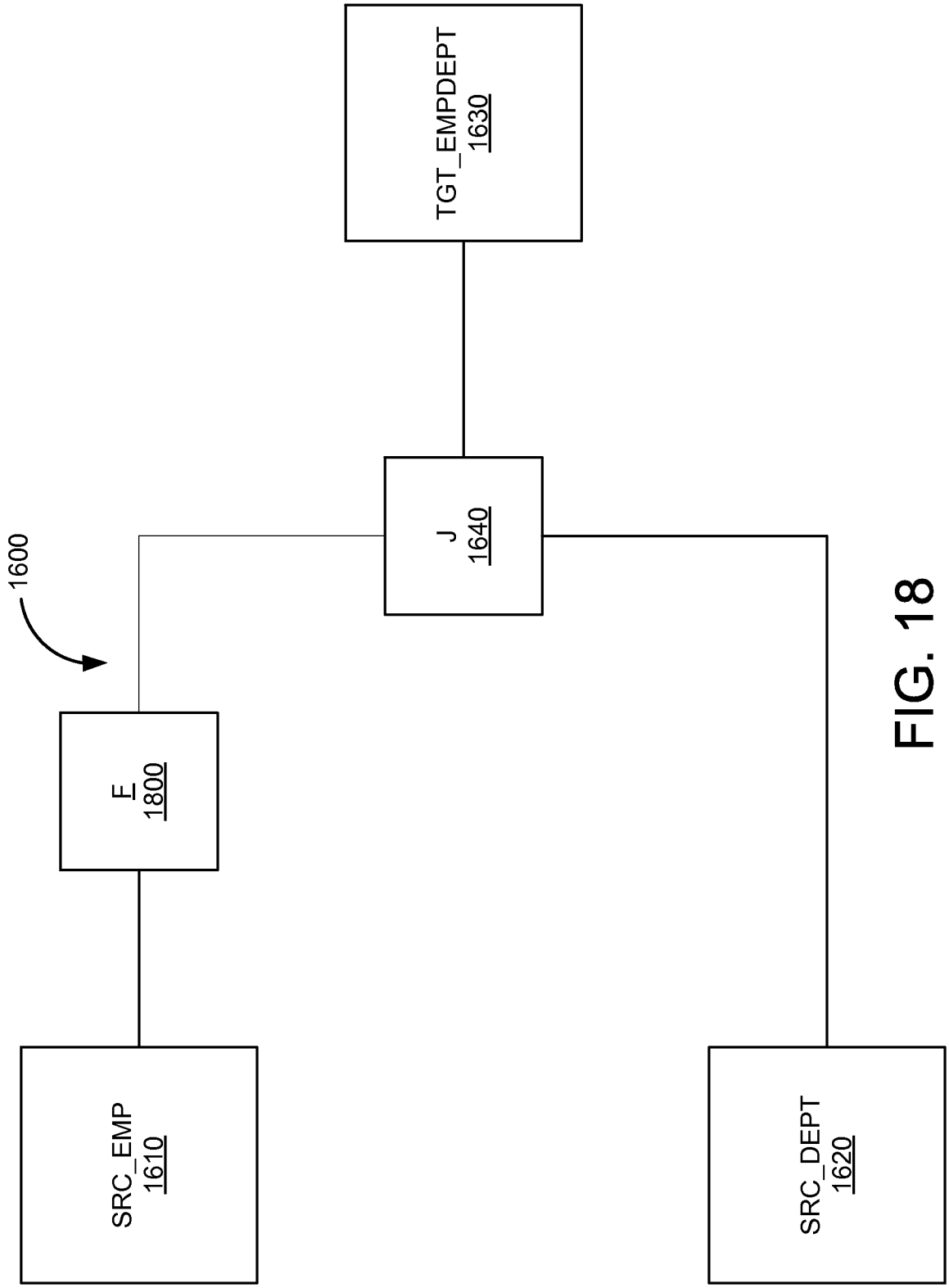

FIG. 18

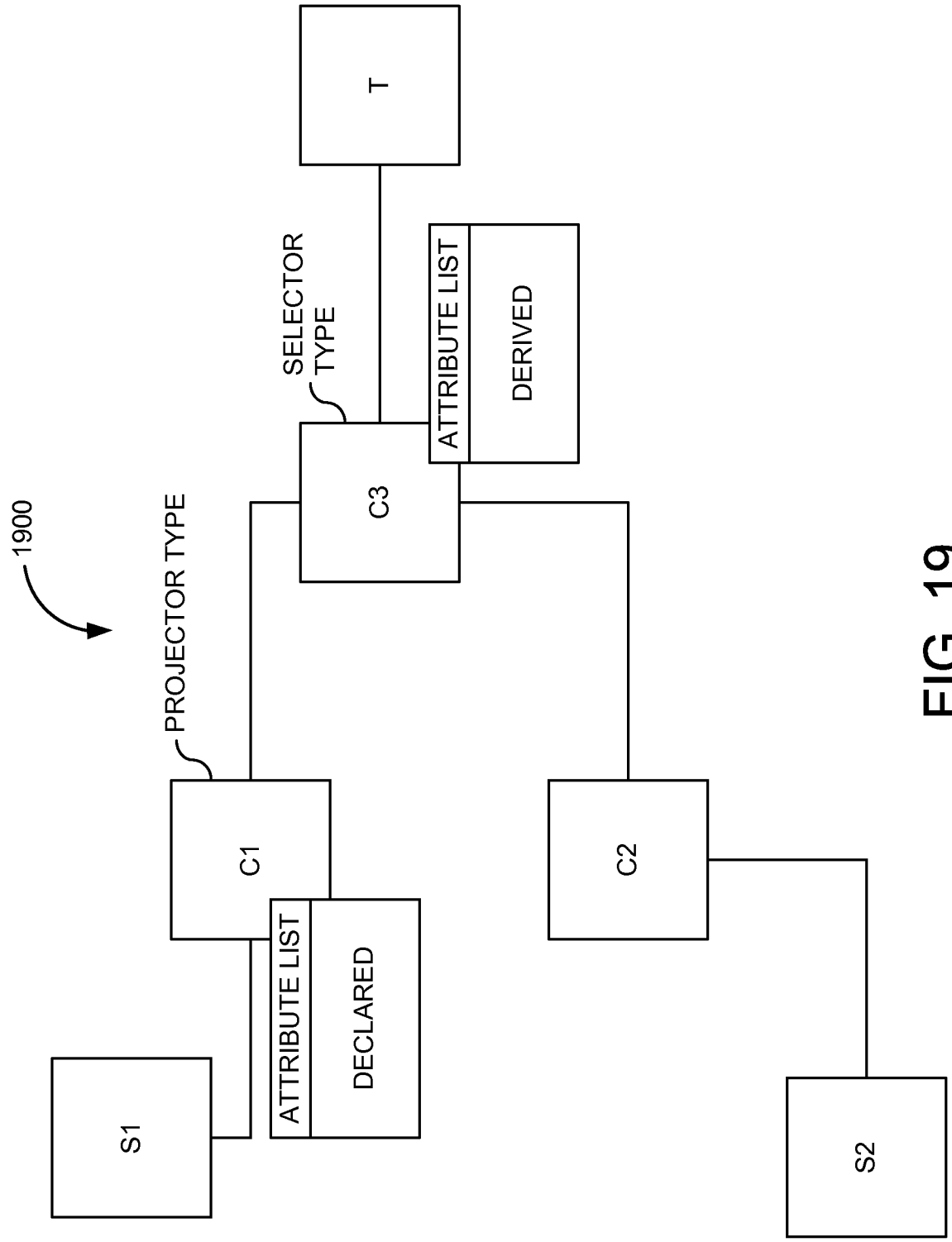
1900
PROJECTOR TYPE
SELECTOR TYPE
T
C3
ATTRIBUTE LIST
DERIVED
C1
ATTRIBUTE LIST
DECLARED
C2
S1
S2

FIG. 19

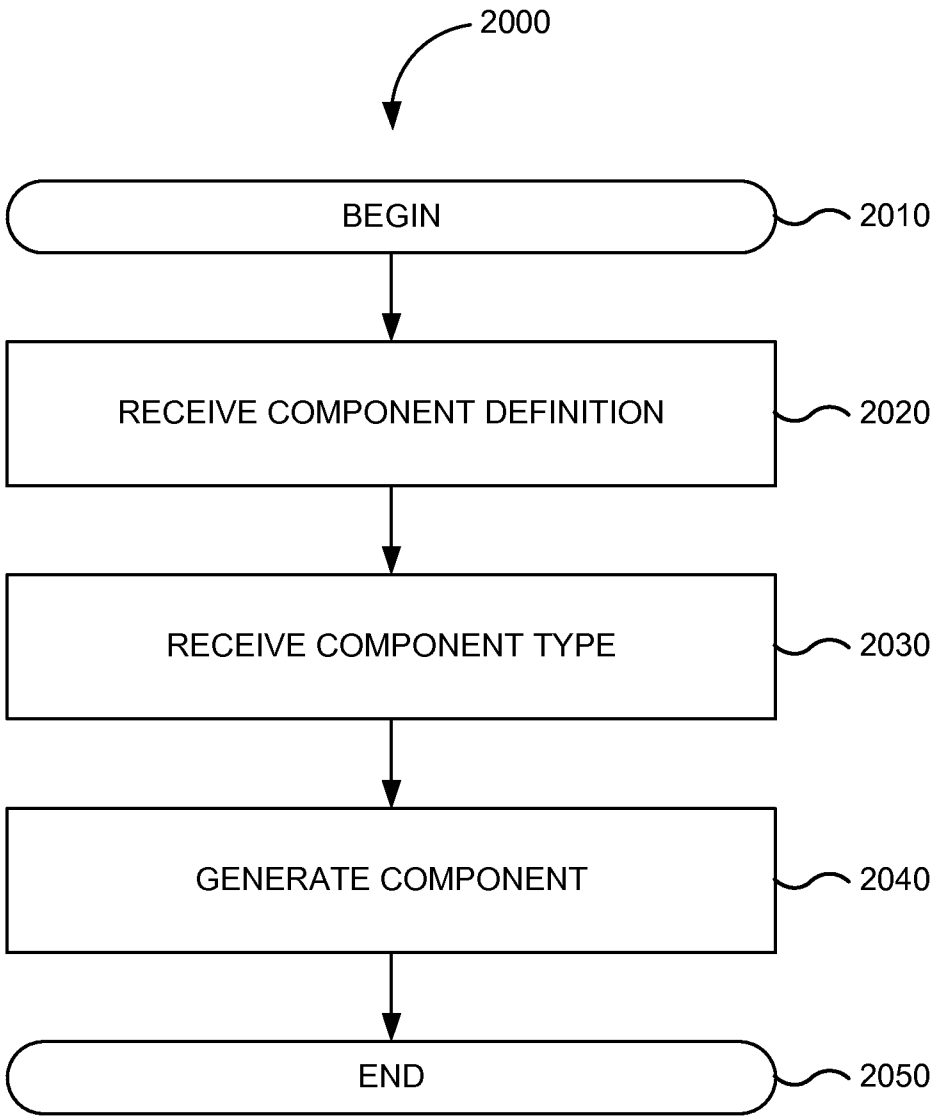
2000
BEGIN
2010
RECEIVE COMPONENT DEFINITION
2020
RECEIVE COMPONENT TYPE
2030
GENERATE COMPONENT
2040
END
2050

FIG. 20

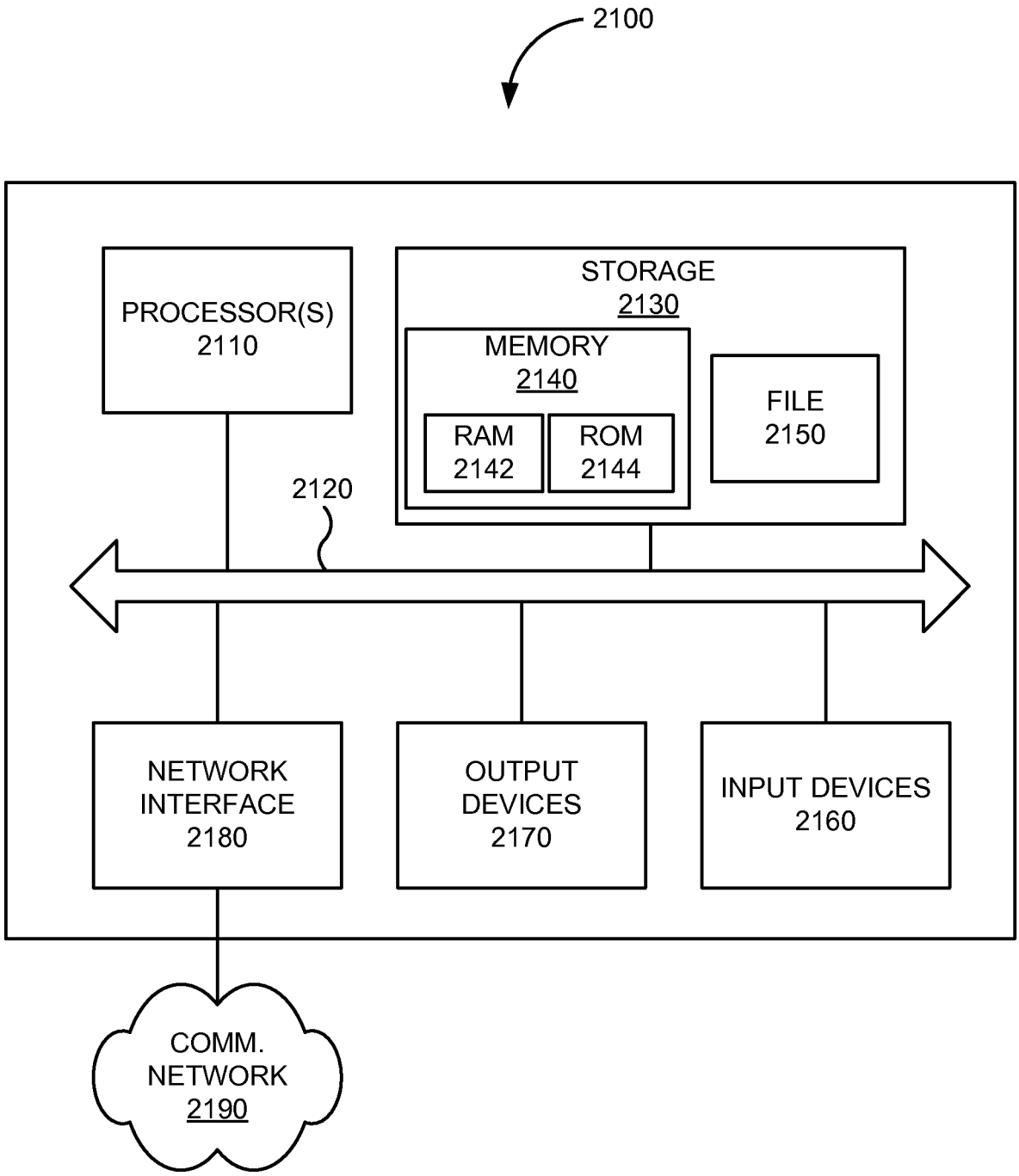
2100
PROCESSOR(S)
2110
STORAGE
2130
MEMORY
2140
RAM
2142
ROM
2144
FILE
2150
2120
NETWORK
INTERFACE
2180
OUTPUT
DEVICES
2170
INPUT DEVICES
2160
COMM.
NETWORK
2190

FIG. 21

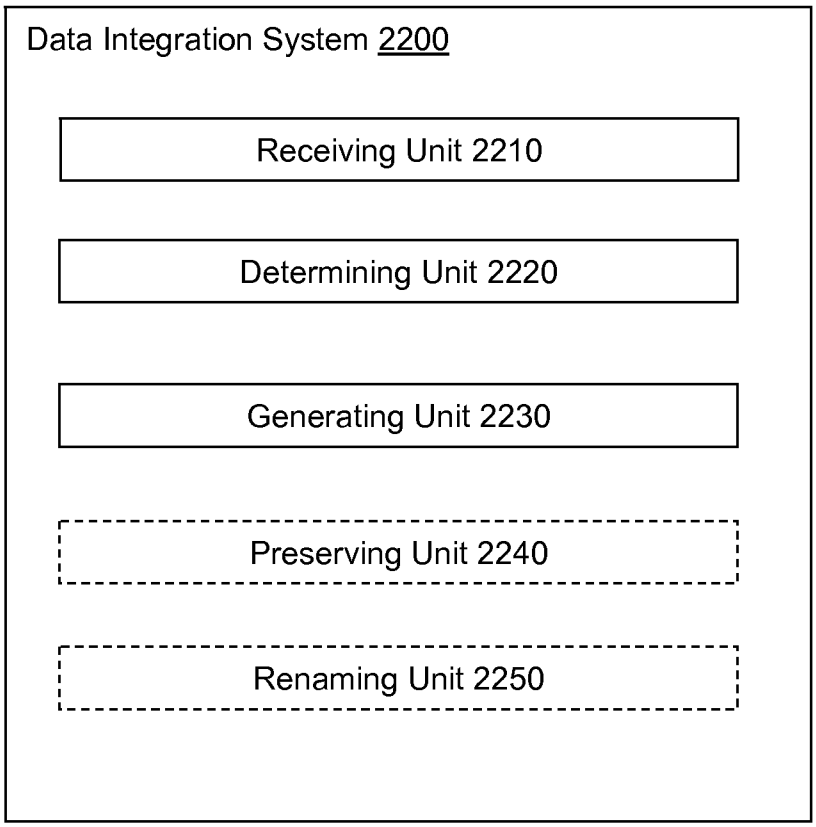
Data Integration System 2200
Receiving Unit 2210
Determining Unit 2220
Generating Unit 2230
Preserving Unit 2240
Renaming Unit 2250

FIG. 22

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/031921

A. CLASSIFICATION OF SUBJECT MATTER

G06F 19/00(2011.01)i, G06F 9/44(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 19/00; G06F 9/44; G06F 17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & keywords: data mapping, attributes, specifying, logical design

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2013-0103705 A1 (SUSAN MARIE THOMAS) 25 April 2013 See paragraphs [0075]-[0086]; and figures 1, 3A.	1-3,11-13,21
A	US 2008-0281849 A1 (KAZUO MINENO) 13 November 2008 See paragraphs [0094]-[0096]; and figure 12.	1-3,11-13,21
A	US 2005-0050068 A1 (ALEXANDER VASCHILLO et al.) 3 March 2005 See paragraphs [0037]-[0044]; and figures 1-2.	1-3,11-13,21
A	US 2011-0295792 A1 (ALEXTAIR MASCARENHAS et al.) 1 December 2011 See paragraphs [0049]-[0052]; and figure 5.	1-3,11-13,21
A	US 2012-0096426 A1 (SERGHEI SARAFUDINOV) 19 April 2012 See paragraphs [0028]-[0042]; and figures 1-8.	1-3,11-13,21

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
31 July 2014 (31.07.2014)	**31 July 2014 (31.07.2014)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701,
Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
JI, Jeong Hoon
Telephone No. +82-42-481-5688

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/031921

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☒ Claims Nos.: 4-10,14-20
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required addtional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fees, this Authority did not invite payment of any additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2014/031921

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2013-0103705 A1	25/04/2013	US 2007-203923 A1	30/08/2007
		US 8307012 B2	06/11/2012
US 2008-0281849 A1	13/11/2008	US 2012-323841 A1	20/12/2012
		US 8280840 B2	02/10/2012
		US 8639670 B2	28/01/2014
		WO 2007-083371 A1	26/07/2007
US 2005-0050068 A1	03/03/2005	CN 100468396 C0	11/03/2009
		CN 1604082 A	06/04/2005
		EP 1519266 A2	30/03/2005
		EP 1519266 A3	26/12/2007
		JP 04847689 B2	28/12/2011
		JP 2005-327232 A	24/11/2005
		KR 10-1159311 B1	22/06/2012
		KR 10-2005-0022272 A	07/03/2005
		US 7739223 B2	15/06/2010
US 2011-0295792 A1	01/12/2011	CN 102918530 A	06/02/2013
		EP 2577507 A2	10/04/2013
		JP 2013-531844A	08/08/2013
		WO 2011-149666 A2	01/12/2011
		WO 2011-149666 A3	19/01/2012
US 2012-0096426 A1	19/04/2012	None	